
AI magazine
Volume 35 Number 4
Winter 2014
A.I.
SPACE
Applications

BE A PART OF AN EXCEPTIONAL PARTNERSHIP

IHMC is re-thinking the relationship between humans and machines, defining new directions in human-centered computing by linking cutting-edge research into a new alliance of AI, robotics, biomedical research, engineering, and social sciences.

From robotic walkers to cognitive orthotics, from autonomy research to human performance enhancement, from new agent architectures to natural language understanding, IHMC research programs are exploring new ways to amplify human intelligence, awareness, and performance.

TULANE UNIVERSITY is one of the nation's most prestigious educational and research institutions. Founded in 1834 in New Orleans, Tulane offers degrees in architecture, business, law, liberal arts, medicine, public health and tropical medicine, the sciences and engineering, and social work.

The newly established computer science department intends to be a leading program with an innovative curriculum that is known for education and research between computer science and other disciplines.

ihmc
FLORIDA INSTITUTE FOR HUMAN & MACHINE COGNITION

Tulane University

JOIN US. *Seeking researchers who have established themselves as leaders in AI (with a particular interest in machine learning) who are interested in building a group and a new academic program. All applicants should apply electronically via: apply.interfolio.com/27214*

For more information on IHMC please explore **ihmc.us**

For more information on Tulane University please explore **tulane.edu**

AI magazine

VOLUME 35, NUMBER 4 Winter 2014 ISSN 0738-4602

Cover: *AI in Space by James Gary, Brooklyn, New York.*

The guest editors for the Space Applications of AI articles in this issue are Steve Chien and Robert Morris

SPECIAL ARTICLES ON SPACE APPLICATIONS OF AI

3 **Editorial Introduction: Space Applications of Artificial Intelligence**
Steve Chien, Robert Morris

7 **Automated Scheduling for NASA's Deep Space Network**
Mark D. Johnston, Daniel Tran, Belinda Arroyo, Sugi Sorensen, Peter Tay, Butch Carruth, Adam Coffman, Mike Wallace

26 **Leveraging Multiple Artificial Intelligence Techniques to Improve the Responsiveness in Operations Planning: ASPEN for Orbital Express**
Russell Knight, Caroline Chouinard, Grailing Jones, Daniel Tran

37 **Enhanced Telemetry Monitoring with Novelty Detection in the Developing World**
José Martínez Heras, Alessandro Donati

47 **Science Autonomy for Rover Subsurface Exploration of the Atacama Desert**
David Wettergreen, Greydon Foil, Michael Furlong, David R. Thompson

61 **Multirobot Coordination for Space Exploration**
Logan Yliniemi, Adrian K. Agogino, Kagan Tumer

ARTICLES

75 **A Review of Real-Time Strategy Game AI**
Glen Robertson, Ian Watson

105 **Power to the People: The Role of Humans in Interactive Machine Learning**
Saleema Amershi, Maya Cakmak, W. Bradley Knox, Todd Kulesza

COMPETITION REPORT

121 **The Dialog State Tracking Challenge Series**
Jason D. Williams, Matthew Henderson, Antoine Raux, Blaise Thomson, Alan Black, Deepak Ramachandran

COLUMN

121 **ACTIVE-ating Artificial Intelligence: Integrating Active Learning in an Introductory Courses**
Marie desJardins

DEPARTMENTS

127 **AAAI News**

112 **AAAI Conferences Calendar**

AI magazine

An Official Publication of the Association for the Advancement of Artificial Intelligence

aimagazine.org

ISSN 0738-4602 *(print)* ISSN 2371-9621 *(online)*

Submissions

Submissions information is available at http://aaai.org/ojs/index.php/aimagazine/information/authors. Authors whose work is accepted for publication will be required to revise their work to conform reasonably to *AI Magazine* styles. Author's guidelines are available at aaai.org/ojs/index.php/aimagazine/about/submissions#authorGuidelines. If an article is accepted for publication, a new electronic copy will also be required. Although *AI Magazine* generally grants reasonable deference to an author's work, the *Magazine* retains the right to determine the final published form of every article.

Calendar items should be posted electronically (at least two months prior to the event or deadline). Use the calendar insertion form at aimagazine.org. News items should be sent to the News Editor, *AI Magazine*, 2275 East Bayshore Road, Suite 160, Palo Alto, CA 94303. (650) 328-3123. Please do **not** send news releases via either e-mail or fax, and do **not** send news releases to any of the other editors.

Advertising

AI Magazine, 2275 East Bayshore Road, Suite 160, Palo Alto, CA 94303, (650) 328-3123; *Fax* (650) 321-4457. *Web:* aimagazine.org. Web-based job postings can be made using the form at https://www.aaai.org/Forms/jobs-submit.php.

Microfilm, Back, or Replacement Copies

Replacement copies (for current issue only) are available upon written request and a check for $10.00. Back issues are also available (cost may differ). Send replacement or back order requests to AAAI. Microform copies are available from ProQuest Information and Learning, 300 North Zeeb Road, Ann Arbor, MI 48106. Telephone (800) 521-3044 or (734) 761-4700.

Copying Articles for Personal Use

Address Change

Please notify AAAI eight weeks in advance of a change of address. Send old label with new address to AAAI, 2275 East Bayshore Road, Suite 160, Palo Alto, CA 94303. We encourage you to notify us via the web: www.aaai.org/Membership/change-form.php.

Subscriptions

AI Magazine (ISSN 0738-4602) is published quarterly in March, June, September, and December by the Association for the Advancement of Artificial Intelligence (AAAI), 2275 East Bayshore Road, Suite 160, Palo Alto, CA 94303, telephone (650) 328-3123. *AI Magazine* is a direct benefit of membership in AAAI. Membership dues are $145.00 individual, $75.00 student, and $285.00 academic / corporate libraries. Subscription price of $50.00 per year is included in dues; the balance of your dues may be tax deductible as a charitable contribution; consult your tax advisor for details. Inquiries regarding membership in the Association for the Advancement of Artificial Intelligence should be sent to AAAI at the above address.

PERIODICALS POSTAGE PAID at Palo Alto CA and additional mailing offices. *Postmaster:* Change Service Requested. Send address changes to *AI Magazine*, 2275 East Bayshore Road, Suite 160, Palo Alto, CA 94303.

Editorial Introduction

Space Applications of Artificial Intelligence

Steve Chien, Robert Morris

■ *We are pleased to introduce the space application issue articles in this issue of AI Magazine. The exploration of space is a testament to human curiosity and the desire to understand the universe that we inhabit. As many space agencies around the world design and deploy missions, it is apparent that there is a need for intelligent, exploring systems that can make decisions on their own in remote, potentially hostile environments. At the same time, the monetary cost of operating missions, combined with the growing complexity of the instruments and vehicles being deployed, make it apparent that substantial improvements can be made by the judicious use of automation in mission operations.*

The case for increasing the level of autonomy and automation for space exploration is well known. Stringent communications constraints are present, including limited communication windows, long communication latencies, and limited bandwidth. Additionally, limited access and availability of operators, limited crew availability, system complexity, and many other factors often preclude direct human oversight of many functions. In fact, it can be said that almost all spacecraft require some level of autonomy, if only as a backup when communications with humans are not available or fail for some reason.

Increasing the levels of autonomy and automation using techniques from artificial intelligence allows for a wider variety of space missions and also frees humans to focus on tasks for which they are better suited. In some cases autonomy and automation are critical to the success of the mission. For example, deep space exploration may require more autonomy in the spacecraft, as communication with ground operators is sufficiently infrequent to preclude continuous human monitoring for potentially hazardous situations.

Space applications of AI can also be divided in terms of three kinds of operations they support: predictable, unpredictable, and real time (Jónsson et al. 2007). Even predictable operations can be extremely complex — enabling artificial intelligence to play a key role in automation to manage complexity or to assist human decision making. Unpredictability of the operating environment increases requirements on the AI system to appropriately respond in a wide range of situations. Real-time requirements may impose limitations on the amount of reasoning performed by the AI system.

 ISSN 0738-4602

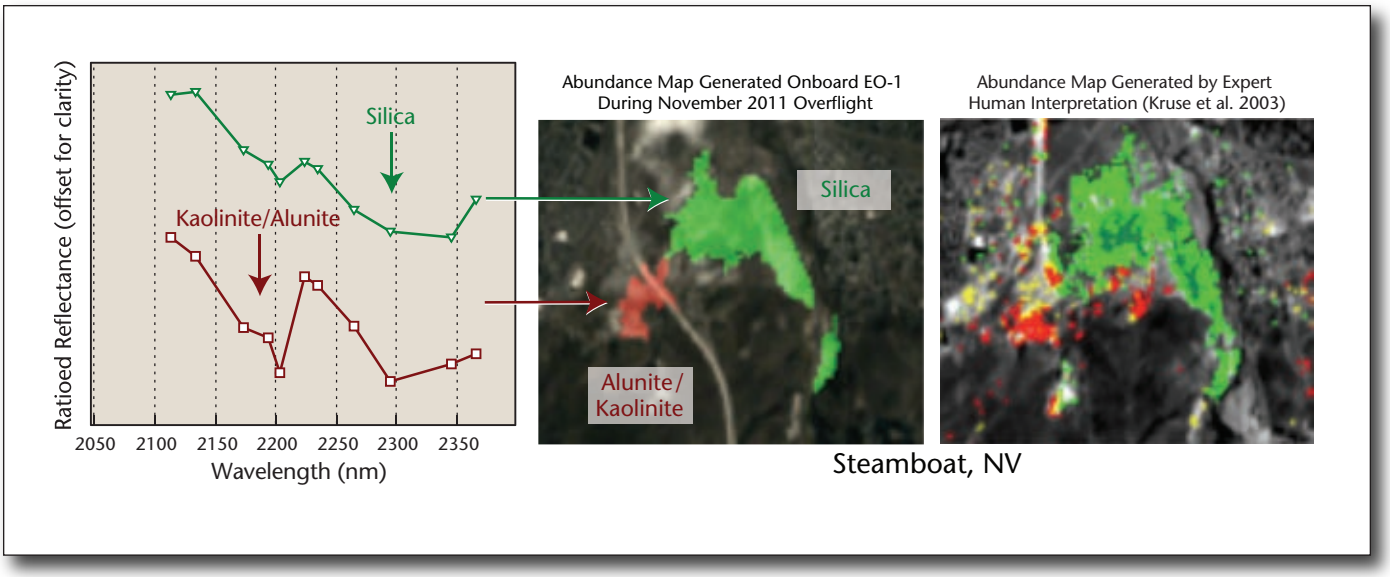


Figure 1. Onboard Spectral Analysis of Imaging Spectroscopy Data During 2011–2012 Demonstrated on EO-1.

Onboard software performed spectral endmember detection and mapping, enabling automatic abundance mapping to reduce data volume by orders of magnitude (Thompson et al 2013). These onboard automatically derived compositional maps (at left) were consistent with prior expert human interpretations (at right).

Of course, deploying large numbers of integrated intelligent agents, each utilizing multiple AI technologies, is the end vision of space AI technologists. The first major step toward this vision was the remote agent experiment (RAX) (Muscettola et al. 1998, Bernard et al. 2000). RAX controlled the Deep Space 1 spacecraft for two periods totaling approximately 48 hours in 1999. RAX included three AI technologies: a deliberative, batch planner-scheduler, a robust task executive, and a model-based mode identification and reconfiguration system.

More recently, the autonomous sciencecraft has controlled the Earth Observing-1 mission for almost 10 years as this article goes to press. This run of operations includes more than 50,000 images acquired and hundreds of thousands of operations goals. The autonomous sciencecraft (Chien et al. 2005a) includes three types of AI technologies: a model-based, deliberative, continuous planner-scheduler (Tran et al. 2004, Rabideau et al. 2004), a robust task executive, and onboard instrument data interpretation including support vector machine-learning derived classifiers (Castano et al. 2006, Mandrake et al. 2012) and sophisticated instrument data analysis (see figure 1) (Thompson et al. 2013b).

Many individual AI technologies have also found their way into operational use. Flight operations such as science observation activities, navigation, and communication must be planned well in advance. AI-based automated planning has found a natural role to manage these highly constrained, complex operations. Early successes in this area include the ground processing scheduling system (Deale et al. 1994) for NASA space shuttle refurbishment and the SPIKE system used to schedule Hubble Space Telescope operations (Johnston and Miller 1994). SPIKE enabled a 30 percent increase in observation utilization (Johnston et al. 1993) for Hubble, a major impact for a multibillion dollar mission. Also impressive is that SPIKE or components of SPIKE have been or are being used for the FUSE, Chandra, Subaru, and Spitzer missions. AI-based planning and scheduling are also in use on the European Space Agency's Mars express and other missions. For a more complete survey of automated planning and scheduling for space missions see Chien et al. (2012a).

In this issue, the article by Mark D. Johnston, Daniel Tran, Belinda Arroyo, Sugi Sorensen, Peter Tay, Butch Carruth, Adam Coffman, and Mike Wallace describes the deep space network (DSN) scheduling engine (DSE) component of a new scheduling system that provides core automation functionality for scheduling of NASA's deep space network, supporting scores of missions with hundreds of tracks every week. The article by Russell Knight, Caroline Chouinard, Grailing Jones, and Daniel Tran describes the application and adaptation of the ASPEN (automated scheduling and planning environment) framework for operations of the Orbital Express (OE) mission.

Because space missions produce enormous petascale data sets, machine learning, data analysis, and event recognition for science and engineering purposes has been another fertile area for application of AI techniques to space applications (Fayyad, Haussler, and Stolorz 1996). An early success was the use

of the decision tree machine-learning techniques in SkiCat (Fayyad, Weir, and Djorgovski 1993) to semi-automatically classify the second Mount Palomar Sky Survey, enabling classification of an order of magnitude greater sky objects than by manual means. Another early advance was the use of Bayesian clustering in the AutoclassAutoClass system (Cheeseman and Stutz 1996) to classify infrared astronomical satellite (IRAS) data. From these beginnings has emerged a plethora of subsequent applications including automatic classification and detection of features of interest in earth (Mazzoni et al. 2007a, 2007b) and planetary (Burl et al. 1998, Wagstaff et al. 2012) remote-sensing imagery. More recently, these techniques are also being applied to radio science signal interpretation (Thompson et al. 2013a).

In this issue the article by José Martínez Heras and Alessandro Donati studies the problem of telemetry monitoring and describes a system for anomaly detection that has been deployed on several European Space Agency (ESA) missions.

Surface missions, such as Mars Pathfinder, Mars Exploration Rovers (MER), and the Mars Science Laboratory (MSL), also present a unique opportunity and challenge for AI. The MER mission uses several AI-related systems: The MAPGEN (Ai-Chang et al. 2004, Bresina et al. 2005) constraint-based planning system for tactical activity planning, the WATCH (Castano et al. 2008) system (used operationally to search for dust devil activity and to summarize information on clouds on Mars.), and the AEGIS system (Estlin et al. 2012) (used for end-of-sol targeted remote sensing to enhance MER science).

Many rover operations, such as long- and short-range traverse on a remote surface; sensing; approaching an object of interest to place tools in contact with it; drilling, coring, sampling, assembling structures in space, are characterized by a high degree of uncertainty resulting from the interaction with an environment that is at best only partially known. These factors present unique challenges to AI systems.

In this issue, the article by David Wettergreen, Greydon Foil, Michael Furlong, and David R. Thompson addresses the use of onboard rover autonomy to improve the quality of the science data returned through better sample selection, data validation, and data reduction.

Another challenge for autonomy is to scale up to multiple assets. While in an Earth-observing context multiple satellites are already autonomously coordinated to track volcanoes, wildfires, and flooding (Chien et al. 2005b, Chien et al. 2012b), these systems are carefully engineered and coordinate assets in rigid, predefined patterns. In contrast, in this issue, the article by Logan Yliniemi, Adrian K. Agogino, and Kagan Tumer tackles the problem of multirobot coordination for surface exploration through the use of coordinated reinforcement learning: rather than being programmed what to do, the rovers iteratively learn through trial and error to take actions that lead to high overall system return.

The significant role of AI in space is documented in three long-standing technical meetings focused on the use of AI in space. The oldest, the International Symposium on Artificial Intelligence, Robotics, and Automation for Space (i-SAIRAS) covers both AI and robotics. I-SAIRAS occurs roughly every other year since 1990 and alternates among Asia, North America, and Europe[1] with 12 meetings to date. Second, the International Workshop on Planning and Scheduling for Space occurs roughly every other year with the first meeting[2] in 1997 with eight workshops thus far. Finally, the IJCAI[3] workshop on AI and space has occurred at each IJCAI conference beginning in 2007 with four workshops to date.

We hope that readers will find this introduction and special issue an intriguing sample of the incredible diversity of AI problems presented by space exploration. The broad spectrum of AI techniques, including but not limited to machine learning and data mining, automated planning and scheduling, multiobjective optimization, and multiagent, present tremendously fertile ground for both AI researchers and practitioners.

Acknowledgements

Portions of this work were carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

Notes

1. See robotics.estec.esa.int/i-SAIRAS.
2. See robotics.estec.esa.int/IWPSS.
3. See ijcai.org.

References

Ai-Chang, M.; Bresina, J.; Charest, L.; Chase, A.; Hsu, J. C.-J.; Jonsson, Ari; Kanefsky, B.; Morris, P.; Rajan, K.; Yglesias, J.; Chafin, B. G.; Dias, W. C.; Maldague, P. F. 2004. Mapgen: Mixed Initiative Planning and Scheduling for the Mars Exploration Rover Mission. *IEEE Intelligent Systems* 19(1): 8–12.

Bresina, J. L.; Jónsson, Ari K.; Morris, P. H.; and Rajan, K. 2005. Activity Planning for the Mars Exploration Rovers. In Proceedings of the Fifteenth International Conference on Automated Planning and Scheduling, 40–50. Menlo Park, CA: AAAI Press.

Bernard, D.; Dorais, G. A.; Gamble, E.; Kanefsky, B.; Kurien, J.; Millar, W.; Muscettola, N.; Nayak, P.; Rouquette, N.; Rajan, K.; Smith, B.; Taylor, W.; and Tung, Y.-W. 2000. Remote Agent Experiment: Final Report. NASA Technical Report 20000116204. Moffett Field, CA: NASA Ames Research Center.

Burl, M. C.; Asker, L.; Smyth, P.; Fayyad, U.; Perona, P.; Crumpler, L.; and Aubele, J. 1998 Learning to Recognize Volcanoes on Venus. *Machine Learning* 30(2–3): 165–194. dx.doi.org/10.1023/A:1007400206189

Castano, A.; Fukunaga, A.; Biesiadecki, J.; Neakrase, L.; Whelley, P.; Greeley, R.; Lemmon, M.; Castano, R.; and Chien, S. 2008. Automatic Detection of Dust Devils and Clouds on Mars. *Machine Vision and Applications* 19(5–6): 467-482. dx.doi.org/10.1007/s00138-007-0081-3

Castano, R.; Mazzoni, D.; Tang, N.; Doggett, T.; Chien, S.; Greeley, R.; Cichy, B.; Davies, A. 2006. Onboard Classifiers for Science Event Detection on a Remote Sensing Spacecraft. In *Proceedings of the 12th ACM International Conference on Knowledge Discovery and Data Mining* (KDD 2006). New York: Association for Computing Machinery. dx.doi.org/10.1145/1150402.1150519

Cheeseman, C., and Stutz, J. 1996. Bayesian Classification (AUTOCLASS): Theory and Results. In *Advances in Knowledge Discovery and Data Mining,* ed. U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy. Menlo Park, CA: AAAI Press.

Chien, S.; Sherwood, R.; Tran, D.; Cichy, B.; Rabideau, G.; Castano, R.; Davies, A.; Mandl, D.; Frye, S.; Trout, B.; Shulman, S.; Boyer, D. 2005a. Using Autonomy Flight Software to Improve Science Return on Earth Observing One. *Journal of Aerospace Computing, Information, and Communication* 2(4): 191–216. dx.doi.org/10.2514/1.12923

Chien, S.; Cichy, B.; Davies, A.; Tran, D.; Rabideau, G.; Castano, R.; Sherwood, R.; Mandl, D.; Frye, S.; Shulman, S.; Jones, J.; Grosvenor, S. 2005b. An Autonomous Earth-Observing Sensorweb. *IEEE Intelligent Systems* 20(3): 16–24. dx.doi.org/10.1109/MIS.2005.40

Chien, S.; Johnston, M.; Policella, N.; Frank, J.; Lenzen, C.; Giuliano, M.; Kavelaars, A. 2012a. A Generalized Timeline Representation, Services, and Interface for Automating Space Mission Operations. Paper presented at the 12th International Conference on Space Operations (SpaceOps 2012), Stockholm, Sweden, 11–15 June.

Chien, S.; McLaren, D.; Doubleday, J.; Tran, D.; Tanpipat, V.; Chitadron, R.; Boonyaaroonnet, S.; Thanapakpawin, B.; and Mandl, D. 2012b. Automated Space-Based Monitoring of Flooding in Thailand. Paper presented at the 11th International Symposium on Artificial Intelligence, Robotics, and Automation for Space (i-SAIRAS 2012), Turin, Italy, September.

Deale, M.; Yvanovich, M.; Schnitzuius, D.; Kautz, D.; Carpenter, M.; Zweben, M.; Davis, G.; and Daun, B. 1994. The Space Shuttle Ground Processing Scheduling System. In *Intelligent Scheduling,* ed. M. Zweben and M. Fox, 423–449. San Francisco: Morgan Kaufmann Publishers.

Estlin, T. A.; Bornstein, B. J.; Gaines, D. M.; Anderson, R. C.; Thompson, D. R.; Burl, M.; Castaño, R.; and Judd, M. 2012. AEGIS Automated Science Targeting for the MER Opportunity Rover. *ACM Transactions on Intelligent Systems and Technology* (TIST), 3(3): Article 50. doi 10.1145/2168752.216876

Fayyad, U.; Haussler, D.; and Stolorz, P. 1996. Mining Scientific Data. *Communications of the ACM* 39(11): 51–57. dx.doi.org/10.1145/240455.240471

Fayyad, U. M; Weir, N.; and Djorgovski, S. G. 1993. SKICAT: A Machine Learning System for Automated Cataloging of Large Scale Sky Surveys. In *Proceedings of the Tenth International Conference on Machine Learning*. San Francisco: Morgan Kaufmann Publishers.

Johnston, M.; Henry, R.; Gerb, A.; Giuliano, M.; Ross, B.; Sanidas, N.; Wissler, S.; and Mainard, J. 1993. Improving the Observing Efficiency of the Hubble Space Telescope. In *Proceedings of the 9th Computing in Aerospace Conference*. Reston, VA: American Institute of Aeronautics and Astornautics. dx.doi.org/10.2514/6.1993-4630

Johnston, M. D., and Miller, G. 1994. Spike: Intelligent Scheduling of Hubble Space Telescope Observations. In *Intelligent Scheduling,* ed. M. Fox and M. Zweben, 391–422. San Francisco: Morgan-Kaufmann Publishers.

Jónsson, A.; Morris, R. A.; and Pedersen, L. 2007. Autonomy in Space: Current Capabilities and Future Challenge. *AI Magazine* 28(4): 27–42.

Kruse, F. A.; Boardman, J. W.; and Huntginton, J. F. 2003. Final Report: Evaluation and Geologic Validation of EO-1 Hyperion. Technical Report, Analytical Imaging and Geophysics LLC, Boulder, CO.

Mandrake, L.; Umaa, R.; Wagstaff, K. L.; Thompson, D.; Chien, S.; Tran, D.; Pappalardo, R. T.; Gleeson, D.; and Castaño, R. 2012. Surface Sulfur Detection via Remote Sensing and Onboard Classification. *ACM Transactions on Intelligent Systems and Technology* (TIST) 3(4): 77. dx.doi.org/10.1145/2337542.2337562

Mazzoni, D.; Garay, M. J.; Davies R.; and Nelson, D. 2007a. An Operational MISR Pixel Classifier Using Support Vector Machines. *Remote Sensing of Environment* 107(1): 149–158. dx.doi.org/10.1016/j.rse.2006.06.021

Mazzoni, D.; Logan, J. A.; Diner, D.; Kahn, R.; Tong, L.; and Li, Q. 2007b. A Data-Mining Approach to Associating MISR Smoke Plume Heights with MODIS Fire Measurements. *Remote Sensing of Environment* 107(1): 138–148. dx.doi.org/10.1016/j.rse.2006.08.014

Muscettola, N.; Nicola, M.; Nayak, P. P.; Pell, B.; and Williams, B. C. 1998. Remote Agent: To Boldly Go Where No AI System Has Gone Before. *Artificial Intelligence* 103(1 (1998): 5–47.

Rabideau, G.; Chien, S.; Sherwood, R.; Tran, D.; Cichy, B.; Mandl, D.; Frye, S.; Shulman, S.; Bote, R.; Szwaczkowski, J.; Boyer, D.; and Van Gaasbeck, J. 2004. Mission Operations with Autonomy: A Preliminary Report for Earth Observing-1. Paper presented at the International Workshop on Planning and Scheduling for Space (IWPSS 2004), Darmstadt, Germany, 23–25 June.

Thompson, D. R.; Brisken, W. F.; Deller, A. T.; Majid, W. A.; Burke-Spolaor, S.; Tingay, S. J.; Wagstaff, K. L.; and Wayth, R. B. 2013a. Real Time Adaptive Event Detection in Astronomical Data Streams: Lessons from the Very Long Baseline Array. *IEEE Intelligent Systems* 29(1): 48–55. 10.1109/MIS.2013.10

Thompson, D. R.; Bornstein, B.; Chien, S.; Schafffer, S.; Tran, D.; Bue, B.; Castano, R.; Gleeson, D.; Noell, A. 2013b. Autonomous Spectral Discovery and Mapping onboard Onboard the *EO-1* Spacecraft. *IEEE Transactions on Geoscience and Remote Sensing* 51(6): 3567–3579. dx.doi.org/10.1109/TGRS.2012.2226040

Tran, D.; Chien, S.; Rabideau, G.; Cichy, B. 2004. Flight Software Issues in Onboard Automated Planning: Lessons Learned on *EO-1*. Paper presented at the International Workshop on Planning and Scheduling for Space (IWPSS 2004), Darmstadt, Germany, 23–25 June.

Wagstaff, K. L.; Panetta, J.; Ansar, A.; Greeley, R.; Hoffer, M. P.; Bunte, M.; and Schorghofer, N. 2012. Dynamic Landmarking for Surface Feature Identification and Change Detection. *ACM Transactions on Intelligent Systems and Technology* 3(3), article number 49.

Steve Chien is head of the Artificial Intelligence Group and a senior research scientist at the Jet Propulsion Laboratory, California Institute of Technology where he leads efforts in automated planning and scheduling for space exploration. He is a four time honoree in the NASA Software of the year competition.

Robert Morris is a senior researcher in computer science in the Exploration Technology Directorate, Intelligent Systems Division at NASA Ames Research Center. Over the years he has been involved in projects applying AI-based planning and scheduling technologies to solving problems in observation planning, navigation planning and autonomous operations of exploration vehicles.

Automated Scheduling for NASA's Deep Space Network

Mark D. Johnston, Daniel Tran, Belinda Arroyo, Sugi Sorensen, Peter Tay, Butch Carruth, Adam Coffman, Mike Wallace

■ *This article describes the Deep Space Network (DSN) scheduling engine (DSE) component of a new scheduling system being deployed for NASA's Deep Space Network. The DSE provides core automation functionality for scheduling the network, including the interpretation of scheduling requirements expressed by users, their elaboration into tracking passes, and the resolution of conflicts and constraint violations. The DSE incorporates both systematic search- and repair-based algorithms, used for different phases and purposes in the overall system. It has been integrated with a web application that provides DSE functionality to all DSN users through a standard web browser, as part of a peer-to-peer schedule negotiation process for the entire network. The system has been deployed operationally and is in routine use, and is in the process of being extended to support long-range planning and forecasting and near real-time scheduling.*

NASA's Deep Space Network (DSN) provides communications and other services for planetary exploration missions as well as other missions beyond geostationary orbit, supporting both NASA and international users. It also constitutes a scientific facility in its own right, conducting radar investigations of the moon and planets, in addition to radio science and radio astronomy. The DSN comprises three antenna complexes in Goldstone, California; Madrid, Spain; and Canberra, Australia. Each complex contains one 70 meter antenna and several 34 meter antennas (figure 1), providing S-, X-, and K-band up- and downlink services. The distribution in longitude enables full sky coverage and generally provides some overlap in spacecraft visibility between the complexes. A more detailed discussion of the DSN and its large antennas can be found in the paper by W. A. Imbriale (2003).

The process of scheduling the DSN is complex and time-consuming. There is significantly more demand for DSN services than can be handled by the available assets. There are numerous constraints on the assets and on the timing of communications supports, due to spacecraft and ground operations rules and preferences. Most DSN users require a

 ISSN 0738-4602

Figure 1. Three of the Deep Space Network 34 Meter Antennas at the Goldstone Deep Space Communications Complex in California.

firm schedule around which to build spacecraft command sequences, weeks to months in advance. Currently there are several distributed teams who work with missions and other users of the DSN to determine their service needs, provide these as input to an initial draft schedule, then iterate among themselves and work with the users to resolve conflicts and come up with an integrated schedule. This effort has a goal of a conflict-free schedule by eight weeks ahead of the present, which is frequently hard to meet in practice. In addition to asset contention, many other factors such as upcoming launches (and their slips) contribute to the difficulty of building up an extended conflict-free schedule.

There have been various past efforts to increase the level of scheduling automation for the DSN (Bell 1993; Biefeld and Cooper 1991; Chien et al. 1997; Fisher et al. 1998; Guillaume et al. 2007; Kan, Rosas, and Vu 1996; Loyola 1993; Werntz, Loyola, and Zendejas 1993). Currently, the DSN scheduling process is centered on the service preparation subsystem (SPS), which provides a central database for the real-time schedules and for the auxiliary data needed by the DSN to operate the antennas and communications equipment (for example, view periods, sequence-of-events files). The current project to improve scheduling automation is designated the service scheduling software, or S^3, which will be integrated with SPS. There are three primary features of S^3 that are expected to significantly improve the scheduling process. (1) Automated scheduling of activities with a request-driven approach (as contrasted with the previous activity-oriented approach that specified individual activities); (2) unifying the scheduling software and databases into a single integrated suite covering real time out through as much as several years into the future; and (3) development of a peer-to-peer collaboration environment for DSN users to view, edit, and negotiate schedule changes and conflict resolutions.

The collaboration environment is described elsewhere (Carruth et al. 2010); this article focuses on the first and second areas and some of their ramifications. (For additional information see Clement and Johnston [2005]; Johnston and Clement [2005]; Johnston et al. [2009]; Johnston et al. [2010].)

The request-driven paradigm shifts the emphasis

Typical number of tracking passes per week	425
Number of users (missions, science users, and maintenance)	37
Typical pass duration	5.25 hours
Assets	12 antennas at 3 sites (to be augmented to 16 by 2020)
Asset loading	~80–95 percent
Scheduling time scale	Preview schedule 17–26 weeks ahead Conflict free 8 weeks ahead

Table 1. Some Characteristics of the DSN Scheduling Problem.

from individual specific resource allocations to a more abstract scheduling request specification or language and on the scheduling algorithms that work with this specification to generate, maintain, and improve the schedule. In the following sections, we first provide some background on the DSN scheduling problem and on the reasons for the request-driven approach taken by S^3. We then briefly describe the scheduling request specification itself, which is how DSN users of S^3 convey their service requests to the system. These requests are processed by the DSN scheduling engine (DSE) to expand into tracking passes and integrate them into an overall schedule, all the while seeking to minimize conflicts and request violations. We conclude with an overall summary and brief description of plans for future development.

Overview of DSN Scheduling

The DSN antennas and supporting infrastructure are heavily used. Characteristics of the network's assets and typical usage are listed in table 1. Currently the DSN supports 37 spacecraft or service users, counting all those with regular requirements for scheduled time on any antenna. The mission users span a wide range of distance and orbit type: high Earth orbit, lunar orbit, solar orbit, probes at Mercury, Venus, Mars, and Saturn (and en route to Jupiter and Pluto/Charon), and to comets and asteroids, out to the two *Voyager* spacecraft in interstellar space. Ground-based users conduct radio science and radio astronomy using the antennas, including coordinated programs with international partners. Other activities that must be scheduled include routine and special maintenance, calibration, engineering, and test activities. The collected set of DSN users imposes a very wide range of usage requirements on the network due to differing designs and operating modes. Some users require occasional contacts of only a few hours per week, but this ranges up to continuous coverage during certain mission phases, such as post-launch and during critical mission events. At the present time, a typical week includes between 400 and 500 scheduled activities on the antennas of the three DSN complexes; a portion of such a schedule is shown in figure 2 in the S^3 web GUI.

Phases of the DSN Scheduling Process

The DSN scheduling process consists of three phases, which do not have sharply defined boundaries. Below we describe these phases as they exist today; later in this article we discuss plans for how they may change in the future.

Long-Range Planning and Forecasting

In today's system, long-range planning is based on user-provided high-level requirements, specified in the form of a spreadsheet that is interpreted by analysts and entered into a database at JPL. The forecast software employs a statistical allocation method (Lacey and Morris 2002) to estimate when these requirements translate into DSN loading over various time frames. Long-range planning has several major purposes: studies and analyses, down time analysis, and future mission analysis.

For planning studies and analyses, periods of particular interest or concern are examined to determine where there is likely contention among missions, for example around launches or critical mission events (maneuvers, planetary orbit insertion or landings), or when construction of a new DSN antenna is under investigation. Down time analysis involves identifying periods of time when necessary antenna or other maintenance can be scheduled, attempting to minimize the impact on missions. For future mission analysis, missions can, in proposal phase, request analysis of their proposed DSN coverage as part of assessing and costing proposals for new missions. The time range for long-range planning is generally six months or more into the future, sometimes as much as years.

Midrange Scheduling

The midrange scheduling phase is when detailed user requirements are specified, integrated, negotiated, and all tracking activities finalized in the schedule. Starting at roughly 4–5 months before execution, users specify their detailed scheduling requirements

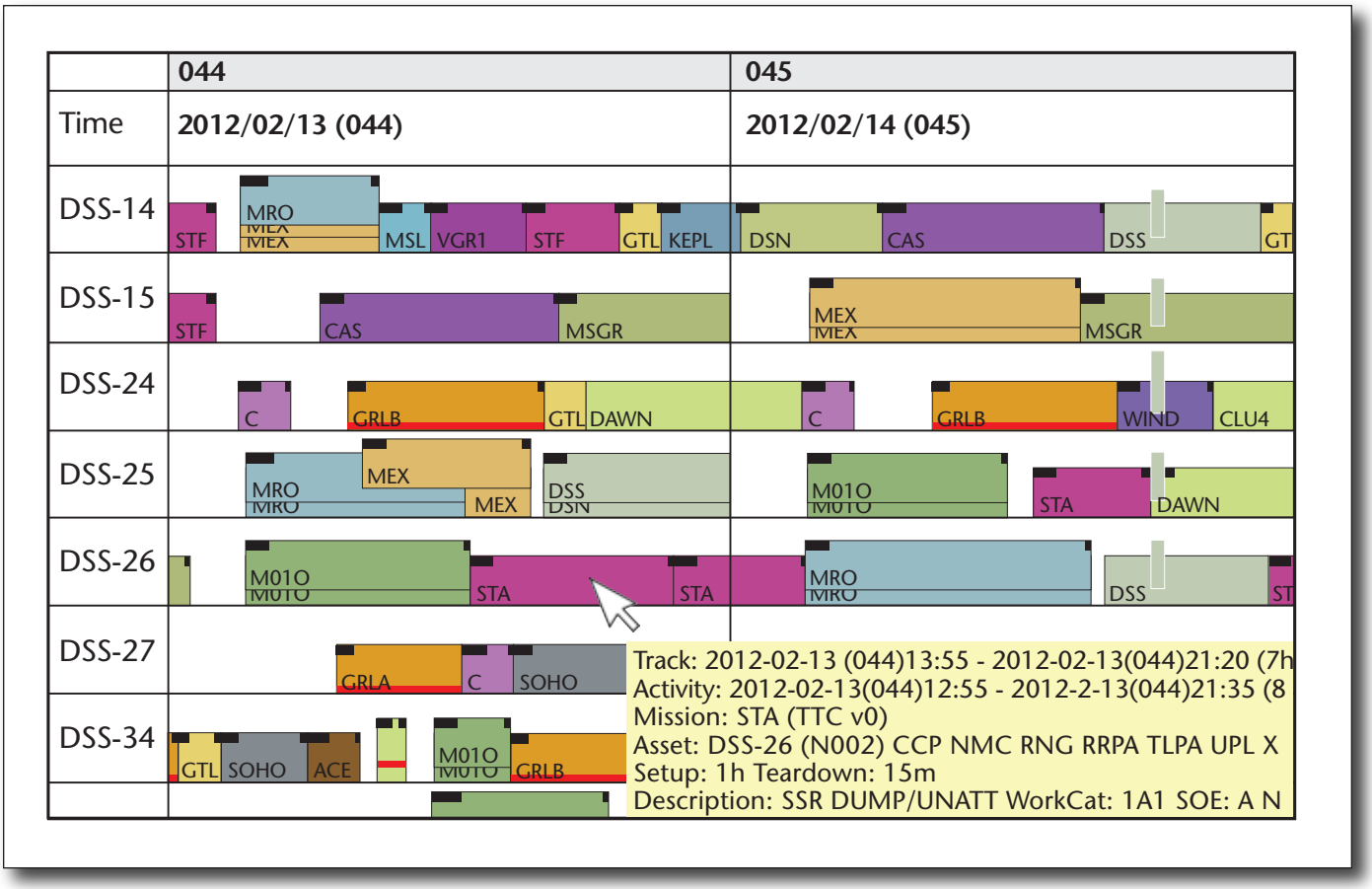


Figure 2. Example of an HTML5 Canvas View of a Portion of the DSN Schedule.

Mousing over a track brings up a transient window with detailed information about the activity (lower right). In this view, different missions are color coded, and setup/teardown is indicated by the black segments at the top left and right of each activity. Each time line represents one of the DSN antennas.

on a rolling weekly basis. These requirements include tracking time and services required, constraining time intervals and relationships (for example, minimum and maximum gaps), visibility constraints, and flexibilities. Further discussion of the nature of these requirements and flexibilities is included in the DSN Scheduling Requests section.

Once the deadline passes and all requirements are in, the full set is integrated into an initial schedule in which conflicts are reduced by taking advantage of whatever flexibilities have been specified. This version of the schedule is extremely heavily overloaded, but it does indicate where there are contentious time periods. These contentious areas shift from week to week depending on critical activities, as well as on the slow drift of visibility intervals with time.

There follows an optimization step where an experienced DSN scheduler interactively edits the schedule and further reduces conflicts by taking advantage of unspecified flexibilities and making further adjustments. At the conclusion of this phase, the schedule usually contains fewer than 30 conflicting sets of activities. It is then released to the scheduling user community who negotiate to reduce conflicts and further optimize coverage for their missions.

It is important to note that, unlike many other scheduling domains, the DSN follows a collaborative approach to developing the conflict-free schedule. DSN users follow a peer-to-peer approach to resolving conflicts. Users create change proposals, which are suggestions as to how different sets of users could modify their tracking passes to resolve conflicts. The affected users can concur or reject these suggestions and counter with suggestions of their own. Over the course of a few weeks, convergence is reached and the schedule reaches a negotiated conflict-free status. Should users not come to agreement among themselves, there is an escalation process to adjudicate irreconcilable conflicts; escalation very rarely occurs in practice. When negotiation concludes, the schedule is conflict free or has only a few waived conflicts for specific reasons. This is considered the negotiated schedule that missions use to plan their integrated ground and spacecraft activities, including the

development of on-board command loads based in part on the DSN schedule.

Following this point, changes to the schedule may still occur, but new conflicts may not be introduced (by policy). There is a continuing low level of no-impact changes and negotiated changes that occur all the way down to real time.

Near Real-Time Scheduling

The near real-time phase of DSN scheduling starts roughly 2–3 weeks from execution and includes the period through execution of all the scheduled activities. Late changes may occur for various reasons (sometimes affecting the midrange phase as well). For example, users may have additional information or late changes to requirements for a variety of reasons; DSN assets (antennas, equipment) may experience unexpected down times that require adjustments to the schedule to accommodate; or spacecraft emergencies may occur that require extra tracking or changes to existing scheduled activities. For many missions that are sequenced well in advance, late changes cannot be readily accommodated.

DSN Scheduling Requests

DSN users represent their needs to the S^3 software system as scheduling requests. Each such request is interpreted by the DSN scheduling engine. The main elements of a scheduling request are service specification, timing constraints, track relationships, priority, preferences, repetitions, and nonlocal time line constraints.

Service Specification

S^3, through the DSE, provides an abstraction level on top of DSN asset specifications that may be referenced by users much more simply than specifying all of the possible options. At the physical level, the spacecraft on-board electronics (frequency band, data rates, encoding), radiated power, distance, along with the DSN antennas, receivers and transmitters, and other equipment, determine what space and ground configurations are feasible. The abstraction level provided in S^3 is called a service alias such that a single service alias encapsulates a wide range of options, preferences, and associated information that is required to schedule the network. For example, some users need the added sensitivity of more than one antenna at a time and so must be scheduled as antennas arrays using two or more antennas at once (as many as four at a time). For navigation data, there are special ranging scenarios that alternate the received signal between the spacecraft and a nearby quasar, over a baseline that extends over multiple DSN complexes. For Mars missions, there is a capability for a single antenna to communicate with several spacecraft at once (called multiple spacecraft per antenna, or MSPA); while more than one at a time may be sending data to Earth, only one at a time may be receiving data sent from Earth.

A more detailed description of service alias functionality is provided in the description of the DSN scheduling engine that follows.

Timing Constraints

Users need a certain amount of communications contact time in order to download data and upload new command loads, and for obtaining navigation data. How this time is to be allocated is subject to many options, including whether it must be all in one interval or can be spread over several, and whether and how it is related to external events and to spacecraft visibility. Among the factors that can be specified in a schedule request are reducible (whether and how much the requested time can be reduced, for example to resolve conflicts); extendable (whether and how much the request time can be extended, should the option exist); splittable (whether the time must be provided in one unbroken track, or can be split into two or more separate tracks); split duration (if splittable, the minimum, maximum, and preferred durations of the split segments; the maximum number of split segments); split segment overlap (if the split segments must overlap each other, the minimum, maximum, and preferred duration of the overlaps); split segment gaps (if the split segments must be separated, the minimum, maximum, and preferred duration of the gaps); quantization (whether scheduled activity times are to occur on 1-minute or 5-minute boundaries); view periods (periods of visibility of a spacecraft from a ground station, possibly constrained to special limits, rise/set, other elevation limits, and possibly padded at the boundaries); and events, which are general time intervals that constrain when tracks may be allocated. Event examples include day of week, time of day (for accommodating shift schedules, daylight, and others); (orbit/trajectory events (occultations, maneuvers, surface object direct view to Earth). Different event intervals may be combined (with optional inversion), and applied to a request.

Track Relationships

In some cases, contacts need to be sufficiently separated so that on-board data collection has time to accumulate data but not overfill on-board storage. In other cases, there are command loss timers that are triggered if the time interval between contacts is too long, placing the spacecraft into safe mode. During critical periods, it may be required to have continuous communications from more than one antenna at once, so some passes are scheduled as backups for others.

Priority

The DSN currently has a priority scheme that ranges from 1–7, with 7 being nominal tracking and 1 representing a spacecraft emergency. Priority is relatively infrequently used, but it does have the effect that the scheduling engine will try to avoid conflicts with higher-priority activities if possible. Depending on their degree of flexibility, missions trade off and com-

promise in order to meet their own requirements, while attempting to accommodate the requirements of others. As noted above, one of the key goals of S^3 is to facilitate this process of collaborative scheduling.

Preferences

Most preferences are incorporated in the service alias and timing requirements described above, but some are directly representable in the scheduling request. For example, users may choose to schedule early, centered, or late with respect to the view period or event timing interval.

Repetitions

One characteristic of DSN scheduling is that, for most users, it is common to have repeated patterns of requests over extended time intervals. Frequently these intervals correspond to explicit phases of the mission (cruise, approach, fly-by, orbital operations). These patterns can be quite involved, since they interleave communication and navigation requirements. S^3 provides for repeated requests, analogous to repeated or recurrent meetings in calendaring systems, in order to minimize the repetitive entry of detailed request information.

Nonlocal Time Line Constraints

Some users have constraints that affect allocations in a nonlocal manner, meaning that an extended time period and possibly multiple activities may have to be examined to tell whether some preferred condition is satisfied. Examples of these constraints include *n* of *m* tracks per week should be scheduled on southern hemisphere tracking stations; *x* hours of tracking and ranging per day must be scheduled from midnight to midnight UTC; the number and timing of tracks in a week should not allow the on-board recorder to exceed its expected capacity.

The DSN Scheduling Engine

The DSE is the component of S^3 responsible for expanding scheduling requests into individual communications passes by allocating time and resources to each; identifying conflicts in the schedule, such as contention for resources and any violations of DSN scheduling rules, and attempting to find conflict-free allocations; checking scheduling requests for satisfaction, and attempting to find satisfying solutions; identifying scheduling opportunities, based on resource availability and other criteria, or meeting scheduling request specifications; and searching for and implementing opportunities for improving schedule quality

Schedule conflicts are based only on the activity content of the schedule, not on any correspondence to schedule requests, and indicate either a resource overload (for example, too many activities scheduled on the available resources) or some other violation of a schedule feasibility rule. In contrast, violations are associated with scheduling requests and their tracks, and indicate that in some way the request is not being satisfied. Conflicts and violations are permitted to exist in the schedule — both are identified by the scheduling engine, recorded in the S^3 database, and made visible to users working with the schedule. The scheduling engine provides algorithms to reduce or eliminate both conflicts and violations where possible, as described below. A block diagram of the DSE is shown in figure 3, showing the overall dependencies of the interface message types on the various software modules.

Architecture

The DSE is based on ASPEN, the planning and scheduling framework developed at Jet Propulsion Laboratory and previously applied to numerous problem domains (Chien et al. [2000]; see also Chien et al. [2012] for a comparison with various time line–based planning and scheduling systems). In the S^3 application there may be many simultaneous scheduling users, each working with a different time segment or different private subset of the overall schedule. This has led us to develop an enveloping distributed architecture (figure 4) with multiple running instances of ASPEN, each available to serve a single user at a time. We use a middleware tier to link the ASPEN instances to their clients, on-board an ASPEN manager application (AMA) associated with each running ASPEN process. A scheduling manager application (SMA) acts as a central registry of available instances and allocates incoming work to free servers. This architecture provides for flexibility and scalability: additional scheduler instances can be brought online simply by starting them up: they automatically register with the singleton SMA process, and are immediately available for use. In addition, each AMA provides a heartbeat message to the SMA every few seconds; the absence of an AMA signal is detected as an anomaly, reported by the SMA, which can automatically start additional AMA instances to compensate.

To roll out new software versions or configuration changes, the SMA can automatically terminate AMAs when they become idle, then start up instances on the new version. This provides uninterrupted user service even as software updates are installed. The SMA also allocates free AMA instances to incoming clients, distributing work over all available host machines and thus balancing the load. The SMA can be configured to automatically start additional AMA instances in case the base set on a host all become busy; in this way, service can gracefully degrade in that all users may see slower response times, but none are locked out of the system entirely. Finally, the SMA process can be restarted, for example, to move it to another host, and upon starting up it will automatically locate and register all running AMA instances in the environment, without interrupting ongoing user sessions.

The DSE communicates with clients using an XML-

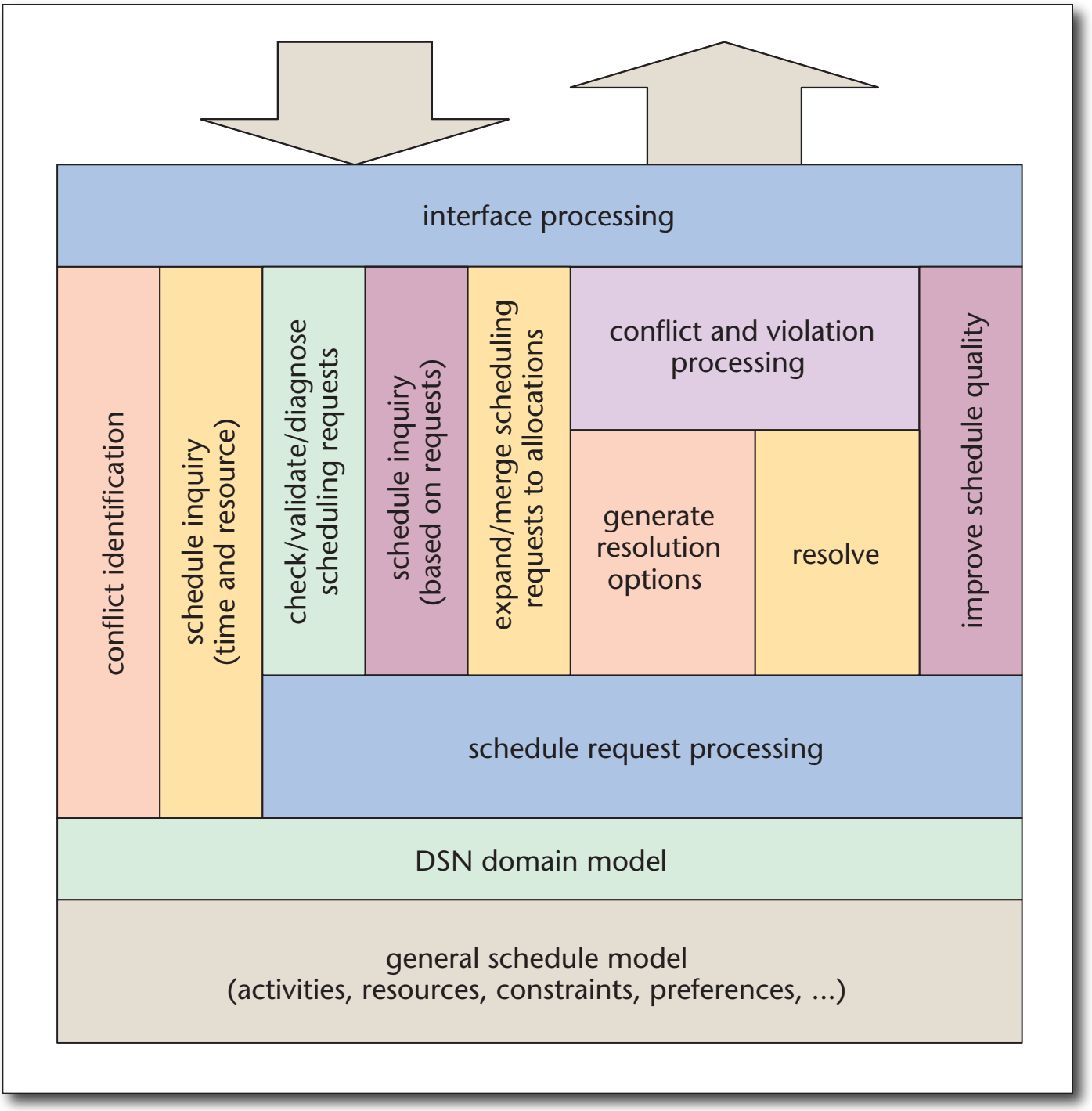


Figure 3. A Block Diagram of the DSE Architecture.

based messaging protocol, similar in concept to HTTP sessions, but with session state maintained by one of the AMA servers, and with responses to time-consuming operations returned asynchronously. Each active user has one or more active sessions, which has loaded all the data related to a schedule that user is working on. This speeds the client-server interaction, especially when editing scheduling requests and activities, when there can be numerous incremental schedule changes.

Next we discuss some of the challenges related to modeling (DSN services, multiple simultaneous spacecrafts, nonlocal time line constraints) and schedule generation and repair algorithms. We also discuss the design of service aliases inasmuch as they underpin all of the DSE functionality.

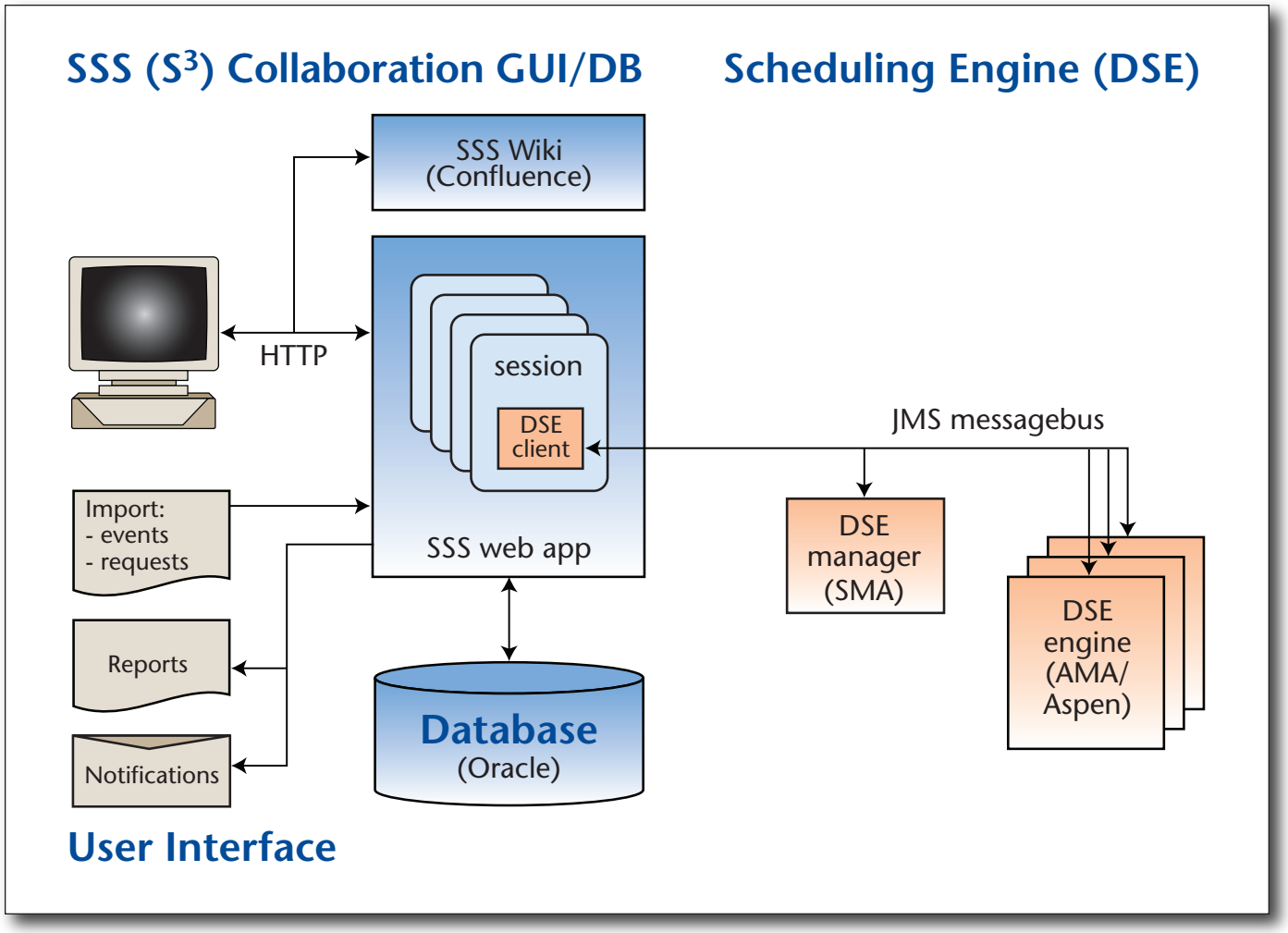


Figure 4. An Overview of the S³ System Architecture.

The DSN scheduling engine manages and provides a set of servers that respond to users' requests for scheduling services through the S³ web application.

Modeling of DSN Services

One of the challenges of modeling the DSN scheduling domain is the wide range of options available for making use of the network. As previously described, one of the primary attributes of a scheduling request is the specification of the DSN services that are needed, which must be transformed into a set of specific resource reservations to satisfy the request. It has been a key element of the DSE design that users can specify their needs at a more general and abstract level, and that the system will translate into the details, ensuring the right antennas and equipment are scheduled. This has the obvious advantage that there is flexibility in the implementation of a request that can be used by the DSN systems, for example, to optimize the schedule or to reschedule on short notice in case assets go down. At the same time, the scheduling system needs to handle a very detailed specification of requested tracking time, down to the selection of individual antennas and equipment types to be reserved. A design to accommodate this spectrum of possibilities has been developed and implemented in the DSE, and is illustrated in figure 5.

Each DSN service user or mission must define one or more service configurations, which are referred to by a name or alias. Each configuration specifies the following information: (1) one or more choices for how antennas and equipment can be allocated to meet the user's DSN requirements; (2) for each choice, which sets of antenna and equipment are acceptable; and (3) for each antenna/equipment combination, what are the default values for associated tracking parameters, such as setup and teardown time before and after the track, the 16-character activity description for the track, a standardized work category used to identify the kind of activity, and if applicable, a specific sequence of events that define all steps that occur during the track.

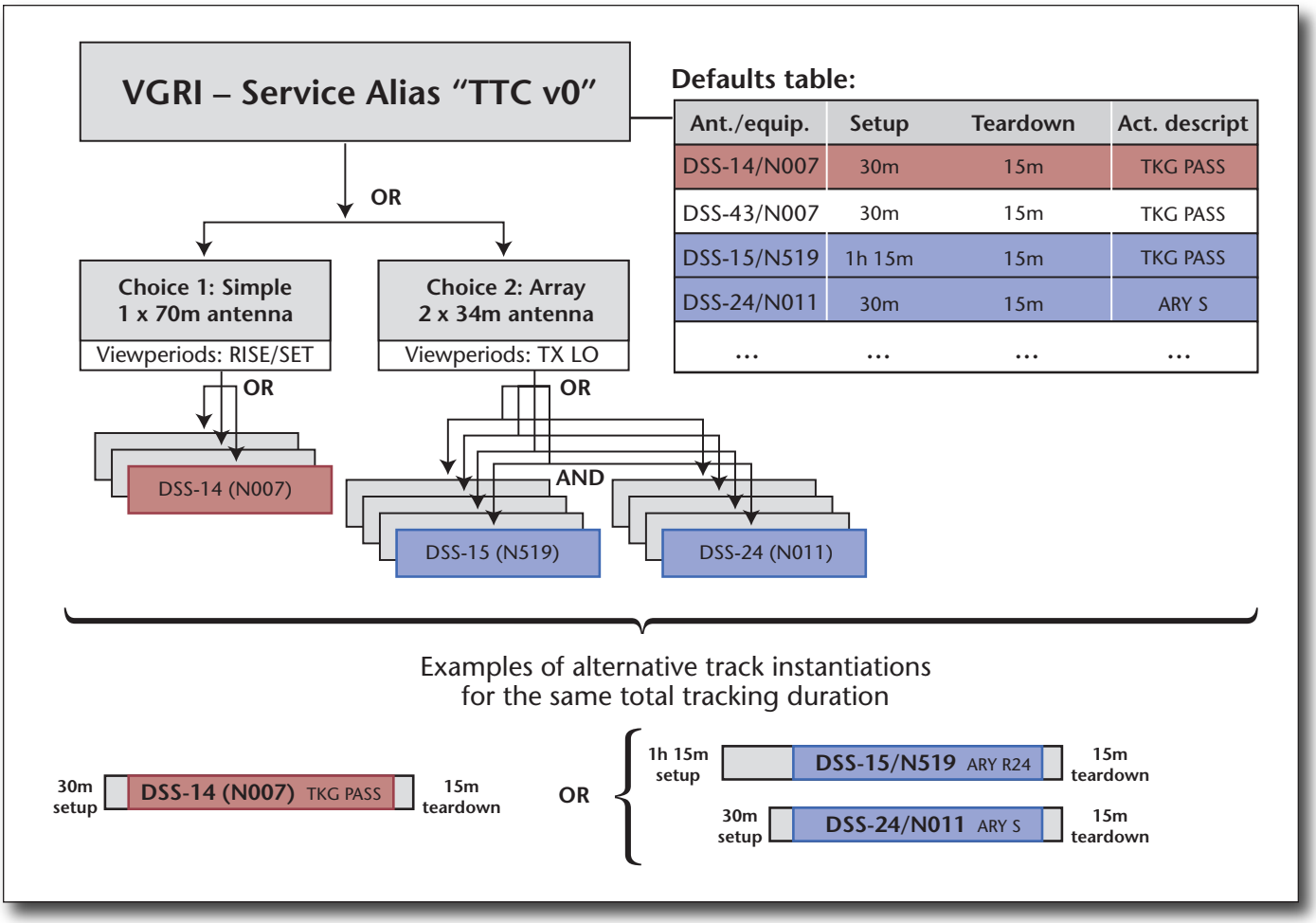


Figure 5. An Illustration of the Structure of a Service Alias Representing a Choice Between a Single Antenna and Multiple Antenna (Array) Implementation of the Same Tracking Duration.

Red highlights the information related to a single-track choice (left) and blue that related to a two-antenna array choice (right). More complex aliases are used to represent up to four station arrays, specialized ranging tracks (DDOR), separate uplink and downlink options for multiple spacecraft tracked all at once, and maintenance activities that affect an entire complex or the entire DSN at once.

A choice within an alias represents a high-level configuration option. For example, some missions may require either a single 70-meter antenna, or two or more arrayed 34-meter antennas. Each of these possibilities corresponds to very different antenna selections, while still satisfying the requirements of the overall service specification. Within a choice, all acceptable sets of antennas/equipment combinations must be specified, in preference order (if applicable). Antenna/equipment combinations within a single antenna choice are in the form of a single list, while those in array choices contain multiple such lists. The same antenna may play different roles within these options, for example as a reference or slave antenna depending on how the equipment is to be configured.

Depending on the nature of the activity, different times must be scheduled for the activity setup (before tracking starts) and teardown (after it completes). Typical setup times are 30 to 90 minutes, while teardown times are usually shorter. The alias definition specifies the default (minimum) setup and teardown time for each antenna/equipment option. In special circumstances these times may be lengthened, but may not be shortened without violating DSN operational rules (and causing a setup or teardown conflict).

Once aliases are defined and validated, their usage in DSE is straightforward. Whenever a user creates a scheduling requirement, a service alias must be specified. The selected alias then determines all the remaining DSN asset requirements and options, while the remainder of the requirement goes on to specify parameters such as timing, duration, and relationships to other tracks. By separating the definition of aliases from their usage, it becomes easier to validate them to ensure that any selection is a legal DSN configuration for that service user.

Most DSN service users will define at least several aliases corresponding to their commonly used scheduling configurations. For example, one alias might specify downlink-only configurations, while another might be used for both downlink and uplink: the latter requires the allocation of transmitters as well as receivers and decoders.

The example illustrated in figure 5 shows how the definition of a service alias for the spacecraft *Voyager 1* encapsulates the alternative options of scheduling on a single 70-meter antenna, or alternatively on a pair of compatible 34-meter antennas. The scheduling user need only specify the service alias name TTC v0, a widely used acronym for telemetry, tracking, and commanding, and the scheduling engine will ensure that any associated activities are compatible with the service alias. Service aliases are versioned over time and can be phased in and out of usage as spacecraft or ground system capabilities change.

In addition to specifying which service alias applies to a given requirement, the DSE provides a capability for overriding the definition of that alias in any requirement in which it is used. An alias override can only restrict the full set of choices allowed by the alias, not add additional ones. As a result, validating the original alias is sufficient to ensure that only legal configurations can be generated by the scheduling system. Examples of possible alias overrides include limits to a single antenna versus an arrayed configuration; limits to one or more DSN complexes (Goldstone, Canberra, or Madrid); limits to a specific antenna subnet (70 meter, 34 meter, and others); and limits to a single specific antenna and equipment combination.

In addition to filtering the set of antenna and equipment choices, users can also override the default values associated with any choice. For example, a particular requirement might need an extended setup time, or customized activity description string that differs from the default. These can be specified using alias overrides.

In addition to antenna and equipment options, certain other attributes of any corresponding activities are also specified by the alias. These include which kind of view period must be used for scheduling, that is, geometrical rise and set versus higher elevation transmitter limits; whether the activity is downlink or uplink only, which is used when scheduling MSPA activities (described in the next section); special activity description suffixes that must be included to indicate certain types of activities; and an effective date and time range.

Service alias definitions are currently captured in XML files that specify all properties of the alias. They are reported in HTML format for users to use and review. A key design feature of the service alias concept in the DSE is that the same XML files are used by the DSE as the domain-specific model of DSN activities and assets, and in the S^3 GUI as the set of all legally selectable choices. Any changes to assets, aliases, or other mission parameters are immediately reflected in the DSE as well as the GUI, without code changes.

Multiple Spacecraft Per Antenna Scheduling

A general capability for the DSN is for a single antenna to communicate with several spacecraft at once (called multiple spacecraft per antenna, or MSPA); while two missions may downlink simultaneously to the antenna, only one may uplink. There are many benefits to sharing antenna time with multiple missions: it provides better utilization of the DSN resources and minimizes antenna setup time needed to support individual tracks. However, there are several drawbacks as well: with multiple missions involved, it increases the complexity of rescheduling of tracks as all missions need to agree to a track change. Also, in the event of a real-time antenna or equipment failure, it increases the number of missions affected. At this time, only Mars missions are able to be part of MSPA groups, though other missions that occupy the same part of the sky, such as Cluster, have also been scheduled as MSPA in the past.

MSPA tracks also have several unique constraints that must be represented within the scheduling engine. No more than two tracks may be downlinking to the antenna simultaneously. No more than one track may be uplinking from the antenna at any time. Only one track per mission can exist within an MSPA group (single track per mission). Only two missions can be scheduled to occur simultaneously. Antenna equipment may be shared between MSPA tracks. Special rules exist for setup and teardown values are used for each MSPA track. These values are dependent on the temporal location of each track. The track with the earliest start time has a different setup value than any tracks that start later. Tracks may be reconfigured midway through execution to uplink/downlink or downlink only tracks. There can only be one reconfiguration per track.

Prior to S^3, MSPA tracks were represented in the same manner as regular tracks. To indicate that a track is a member of an MSPA group, users would manually enter a unique coded string in the track's 16-character activity description field. This string contained required information such as whether the track is uplinking or downlinking, the relative priorities of missions within the group, the reconfiguration time within the track, and the reconfigured equipment state. Using the 16-character activity description field to represent MSPA track details has led to several artificial constraints in the system: a limited number of groups allowed per day, an inability to specify multiple track reconfigurations in one MSPA group, and a limited number of consecutive downlinks.

These limitations have led S^3 to represent MSPA tracks in a different manner. For MSPA-capable mis-

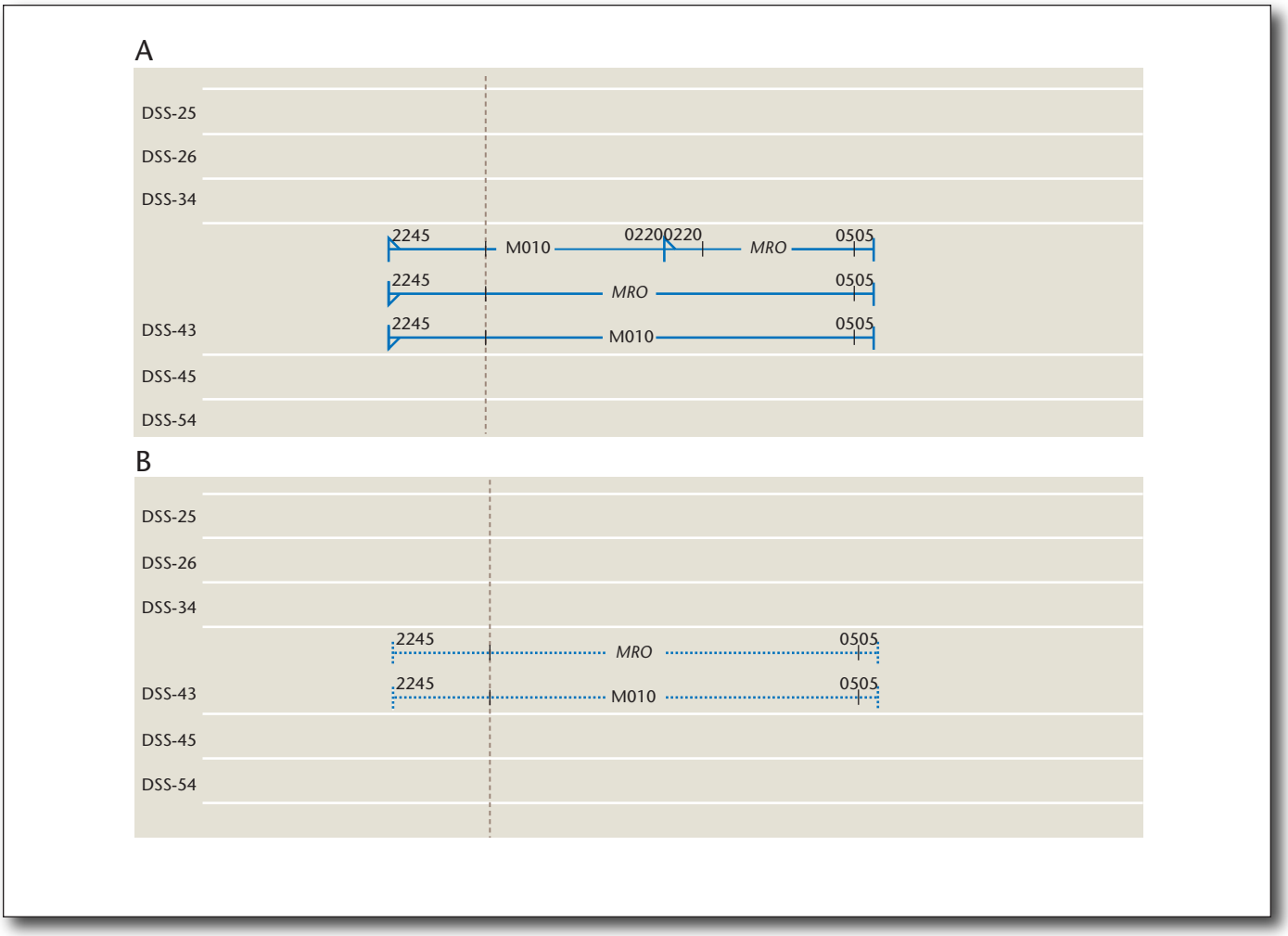


Figure 6. An Example of S³ Scheduled MSPA Activities and Their Corresponding Legacy Tracks.

(a) An example S³ MSPA grouping on DSS-43 where the uplink-only and downlink-only tracks are represented separately. Two spacecraft (MRO and M01O) are downlinking simultaneously while the uplink occurs for M01O at the beginning of the track, then transitions to MRO until the end of the track. The equipment specified for each track represents just the equipment that is needed to support the uplink and downlink services individually. (b) The same MSPA grouping example as in figure 1, but represented in the legacy track view. In this view, the uplink and downlink tracks are merged together and the activity description field contains the track reconfiguration times and supporting equipment needed.

sions, the tracking time required for uplink and downlink may differ. Therefore, S³ services for uplink-only and downlink-only tracks were introduced, specifying only the equipment used for each tracking type. These services are then referenced by the requirements, with different required tracking times specified for uplink and downlink. The engine will then schedule these tracks and attempt to combine them into single tracks where possible. Representing separate uplink and downlink tracking time allows for more flexibility in scheduling individual tracks and removes several of the artificial constraints required by use of the 16-character activity description field. However, to support existing tools and interfaces, a legacy representation of the tracks is still required. In this legacy view, the separate uplink-only and downlink-only tracks are merged together and the activity description fields automatically populated to represent reconfiguration parameters and times. This process is illustrated in figure 6.

The need to merge S³ uplink-only and downlink-only tracks to legacy tracks introduced several issues that needed to be addressed. Given the unique constraints of MSPA tracks and how they are grouped together, the possibility arises that the S³ tracks are organized in a manner such that merging them into legacy tracks is impossible. This is mitigated by ensuring that when the scheduling engine generates tracks for MSPA-capable missions, the tracks are properly grouped together. However, with user control over track parameters, an S³ activity can be easily added, deleted, or modified using the schedule

Request Type	Examples	Parameters
Total time	8 hours of tracking per day 6 hours of uplink tracking each midnight to midnight UTC 24 hours of specific activity types per week summed over four different but related spacecraft	mission(s) service aliases time frame (1 day, 1 week, etc.) min/max tracking times with yellow/red limit
Tracking gaps	6–12 hour gap between tracks, measured midpoint to midpoint Gaps no greater than 8 hours measured EOT to BOT	mission service aliases min track gap max track gap yellow limits measured by (BOT-BOT, EOT-EOT, midtrack to midtrack)
DSN complex distribution	3 of 10 tracks per week must be scheduled at Canberra DSN complex At least one track per week must be scheduled at each DSN complex	mission duration list of (complex, count)
Recorder	Do not exceed on-board recorder volume capacity limit	mission track overhead duration recorder collection rate (X units/s) yellow/red recorder max capacity recorder downlink rates (antenna, downlink rate X units/s) initialization rule

Table 2. Time Line Requirement Types, with Examples and Parameters.

editor. For each group of MSPA tracks, the scheduling engine will report when it is infeasible to generate legacy tracks. When this occurs, the scheduling engine will report a conflict and then output the S^3 tracks as the legacy tracks and assign a special error code in the track's activity description. The types of MSPA conflicts reported are multiple uplinks (more than one track simultaneously uplinking on the same antenna); multiple downlinks (more than two tracks simultaneously downlinking on the same antenna); multiple missions (more than two missions simultaneously tracking); multiple track reconfigurations (more than one track reconfiguration is occurring in the merged uplink-only and downlink-only tracks — this occurs when both the start or end of the tracks differ) (see figure 7); track reconfiguration time (the track reconfiguration time occurs during the pretrack setup time, instead of the during the tracking time); and downlink coverage (an uplink-only track is not fully covered by a downlink-only track). Uplink-only tracks were introduced in S^3 and must be fully merged with downlink-only tracks in order to correctly produce legacy tracks.

In addition, to ensure that the merged legacy tracks meet the service specifications of the user, embedded within the uplink and downlink requirement service aliases are a common set of legal antenna/equipment combinations for the merged tracks. For a legacy track to be considered legal, the merged configuration must be present in the service aliases for that mission. If the antenna/equipment combination is not present, it is reported as a requirement service violation, prompting the user to make the appropriate updates to the tracks. Alternatively, the user may also invoke the scheduling engine to attempt to resolve the violation.

Time Line Constraints and Preferences

The initial development of S^3 has focused on the most frequently encountered types of scheduling request types, which directly affect how DSN antenna allocations are to be constructed. A second broad category of scheduling requirements includes those that indirectly affect allocations in a nonlocal manner. There can be a tradeoff between satisfying these types of requirements, versus the direct requirements noted previously.

We have denoted these types of scheduling requests as time line constraints or preferences, since they are best assessed by considering the overall time line of activities (or subset of activities) for a DSN service user over some time period. Table 2 includes a more detailed list of major time line requirement types and their parameters.

Because these requests have a varying degree of preference, and therefore need to be accessible to the judgement of the scheduling users, we have pursued their incorporation into S^3 in two phases; (1) as inte-

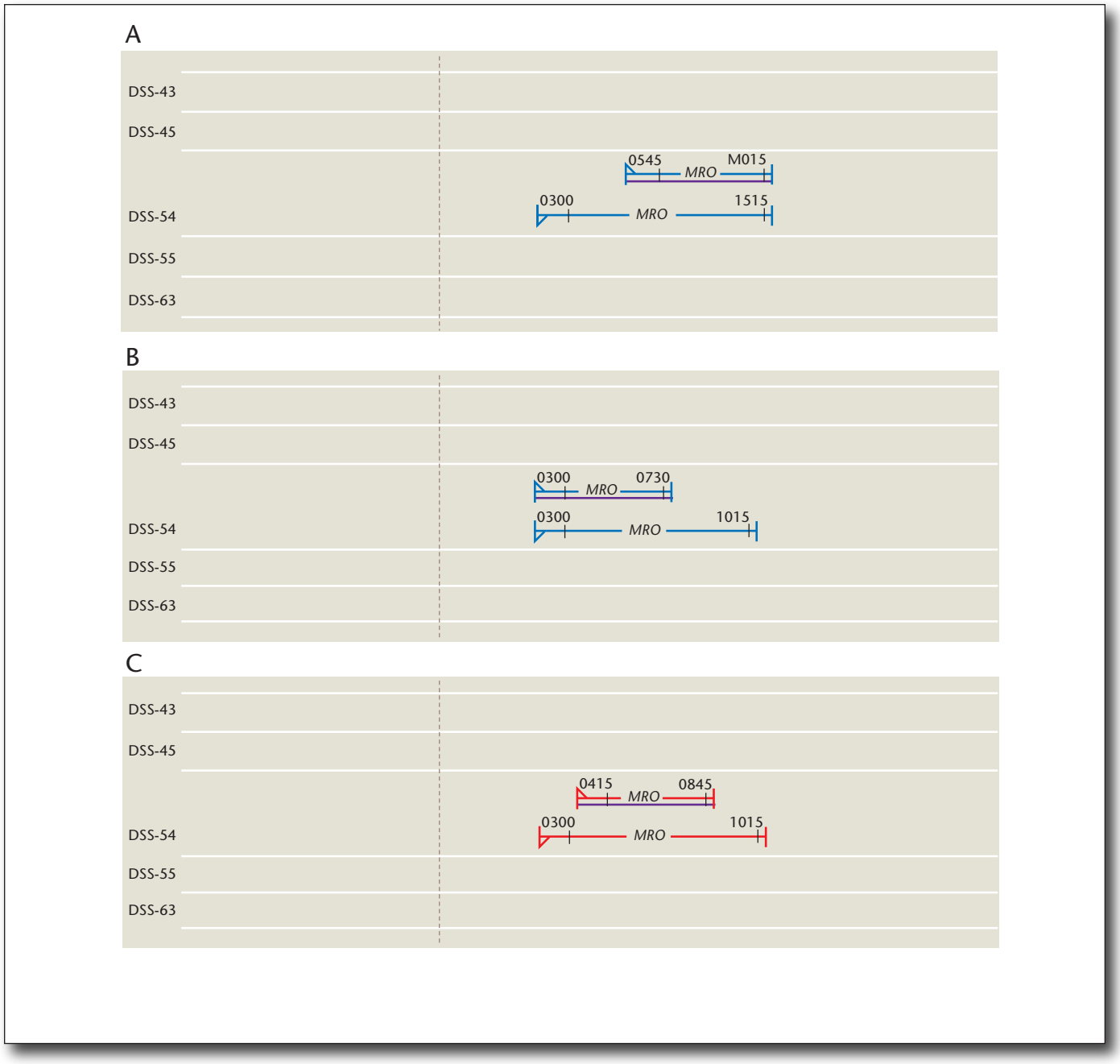


Figure 7. An Example of Multiple Track Reconfiguration Conflicts.

(a) When these two MRO tracks are merged together into a legacy track, the track will start as downlink only and be reconfigured midway at 0545 to uplink and downlink. (b) When these two MRO tracks are merged together into a legacy track, the track will start as uplink and downlink and be reconfigured midway at 0730 to downlink only. (c) An example of a multiple track reconfiguration conflict. If the two tracks are merged together into a legacy track, it would begin as downlink only, reconfigure to uplink and downlink at 0415, and then reconfigure again at 0845 to downlink only. There is a limit of only one track reconfiguration for MSPA tracks.

grated with the scheduling system graphical user interface (GUI), for visualization along with the actual schedule itself; and (2) as incorporated into the DSE algorithm set, for invocation as strategies or heuristic repair and rescheduling options that can be included or not into the normal scheduling process

Integration with the S^3 GUI has built upon the deployed S^3 HTML5 canvas-based GUI (see figure 2), which has enabled the rapid extension of the GUI to additional visualization elements. Examples of the visualization of each of the major categories of time line requirements follow.

The total time requirement applies to about 25 percent of the DSN user set, but over a wide range of time scales, from a full week on down to a fraction of a single day. An example for the GRAIL A/B mission (two spacecraft in lunar orbit) is shown in figure 8a.

The tracking gaps time line requirement applies to about a third of the DSN user set. In some cases, the gaps of concern are only for certain activity types, as illustrated in figure 8b where gaps are only significant between adjacent ranging passes.

About 20 percent of users have DSN complex distribution requirements, but this varies depending on the phase of the mission. These requirements are typically driven by navigation considerations, where it is important to have ranging data from widely separated baselines in order to reduce ephemeris errors. Examples are shown in figure 8a–c, where satisfaction or violation of the distribution requirement is clearly visible.

While most missions have on-board recorders, only a handful can potentially be modeled simply enough to include in the early stages of DSN scheduling. For those missions with uniform data collections rates and well-defined downlink rules, the recorder time line requirement can provide early visibility into recorder capacity and how it is affected by specific scheduling choices. An example is shown in figure 8c for the STEREO A/B spacecraft.

By providing a highly visual view of these time line constraints and preferences, users who are working on schedule changes to resolve conflicts can immediately see whether their proposed changes would introduce any violations. Presently, many scheduling users have custom scripts that they use to evaluate proposals from other users, but by providing for common models and visibility, feedback can be provided much more rapidly. This feedback has the potential to reduce the overall negotiation process effort and duration.

Overview of Scheduling Strategies

There are a few basic design principles around which the DSE algorithms have been developed, derived from the role of the DSE as the provider of intelligent decision support to DSN schedulers. In support of schedule repair and negotiation, it is critically important that the DSE follow a no surprises paradigm, that is, no unexpected schedule changes (all changes to the schedule must be requested, explicitly or implicitly, and the same sequence of operations on the same data must generate the same schedule) and even for infeasible scheduling requests, attempt to return something reasonable in response, possibly by relaxing aspects of the request; along with a diagnosis of the sources of infeasibility, this provides a starting point for users to handle the problem

In contrast to this mode of operation is an autogeneration phase of the scheduling process where the goal is to integrate scheduling requests from all users. The result is an initial schedule with minimal conflicts and violations to serve as a starting point for collaborative conflict resolution. In this mode, maintaining schedule stability is not an objective, and a much broader range of changes to the scheduled activities is allowable, provided that overall conflicts are reduced. The DSE supports both modes of operation with a portfolio of algorithms that can be invoked by the S^3 system for autogeneration, or by end users when working on specific conflicted portions of the schedule.

Expanding Requests to Tracks

The initial layout algorithm is the primary algorithm users invoke to generate tracks to satisfy the specifications of the request. It is also used to remove any existing tracks and regenerate them around whatever other activities already exist in the schedule. The algorithm consists of a series of systematic search stages over the legal track intervals, successively relaxing request constraints at each stage if no solution is found. The systematic search algorithm is a depth-first search algorithm over the space of available antenna/equipment start times and durations for each scheduling request. The set of legal antenna/equipment for scheduling is defined in the request service alias specification, while the search space of legal start times and durations is defined by the request quantization value (usually 5 minutes).

The successive relaxation of constraints allow for tracks to be generated even though the scheduling request may be infeasible (in isolation or within the context of the current schedule), and provides the user a starting point to make corrective changes. These changes may range from modifying the scheduling request to introduce more tracking flexibility, to contacting other mission schedulers to negotiate different request time opportunities. One of the limitations of the initial layout algorithm is its ability to schedule collections of requests associated with track relationships. As it iterates over these requests, tracks may be generated without regard to the feasibility of generating tracks for the future requests in the collection. As a result, it is prone to creating violations for users whose requests are highly interconnected.

Relaxation proceeds in two stages, based on schedule content, then on constraint parameters. In the first phase, if no conflict-free allocation can be found, the engine successively ignores lower priority activities, then those of equal priority, and finally all preexisting activities. If there is still no satisfying allocation, then requirement parameters are relaxed in the following order: (1) timing relationships, (2) gap and overlap parameters for split tracks, and (3) constraining event windows. Ultimately, only antenna-to-spacecraft visibility intervals are considered and an activity of the specified duration is created to overlap one of these.

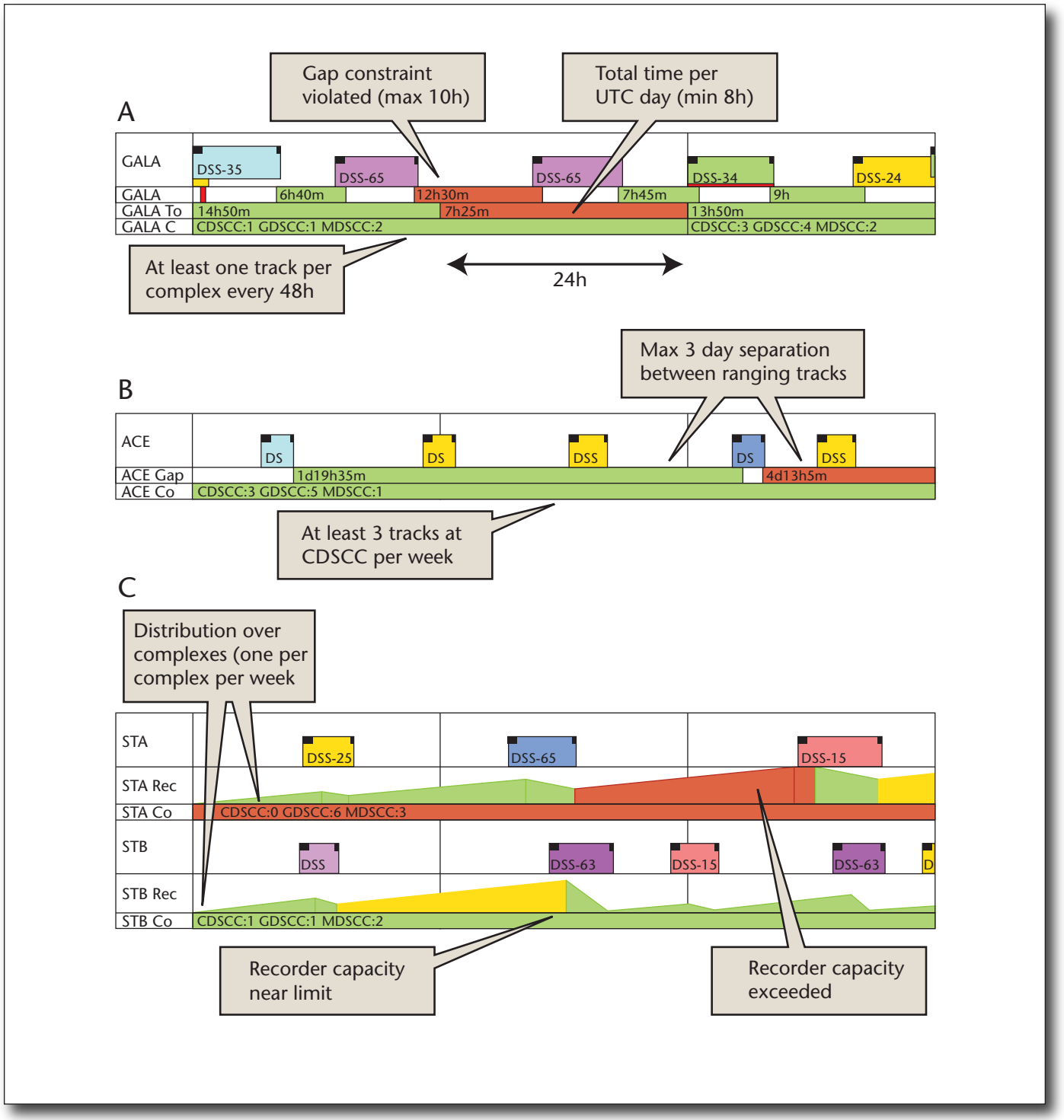


Figure 8. Time Line Constraints for Three Representative Spacecraft, Depicted in the S³ Scheduling HTML5 GUI.

(a) Example of multiple time line requirements applied to a single spacecraft, here GRAIL A, one of a pair of lunar orbiters. There is a gap constraint and a minimum tracking time constraint in a 24-hour UTC day (both violated and colored red); there is also a requirement to track on all three DSN complexes within a 48-hour period (satisfied). (b) Example of a gap constraint between ranging passes only, that is, ignoring the intervening tracking passes. In this example, the second maximum gap requirement has been violated and the resulting interval is colored red. (c) Example of a recorder time line constraint applied to the STEREO A/B mission pair, showing the violation of the constraint in an interval where the accumulated data would exceed the recorder capacity. Note that the recorder volume drops more quickly when a 70-meter contact (such as DSS-63) is scheduled, due to the higher downlink data rate. The STEREO spacecraft also have a requirement to schedule at least one track per week at each complex, here satisfied only for STEREO B.

STN Scheduling

To address the limitations of the initial layout algorithm with interconnected requests, early work has begun using a simple temporal network (STN) to generate tracks for a targeted set of DSN users. The algorithm can be described in two parts: pruning of the legal intervals for each request based on the STN, followed by a systematic search of the pruned legal intervals for a solution.

In pruning the legal intervals, an STN is first initialized with track time constraints on the request boundaries and the track relationships. After propagation, the STN is then used to make a first pass at pruning the legal intervals based on the earliest legal start time and the latest legal end time for each request. A second attempt at pruning the legal intervals is performed by adding additional track time constraints to include the earliest start time and latest end time of a request legal interval. We then systematically begin searching for a solution by temporally assigning a track to the each legal interval and including it into the STN. If the STN is inconsistent, we reassign a track into the next legal interval for that request. Once a consistent STN is found, a valid schedule is generated.

Additional work is still required for the STN scheduling algorithm. At present, it is only used for scheduling tracks for a small subset of the DSN users where the requests are tightly connected with timing constraints to two preceding and two following activities, and additionally have irregular and highly restrictive interval constraints. It will also need to be extended to support relaxing specific request constraints to generate some tracks. With the current implementation, if a valid solution cannot be found, no tracks are generated. This is undesirable as it provides no feedback to the user to determine what the problem may be in the requests or schedule.

Repairing Conflicts and Violations in the Schedule

Once an initial schedule has been generated, conflicts and/or violations may exist in the schedule due to the relaxation of constraints (Johnston and Giuliano 2011). The DSE provides schedule repair algorithms to reduce conflicts or violations. These algorithms identify the contributing tracks for each conflict or violation, and run the systematic search algorithm on the request. If a solution is found, the new tracks are accepted. If no solution is found, the original tracks are not modified. Note that conflicts and violations are independent, so there are separate versions provided through the user interface for users to invoke. This algorithm is focused on only modifying requirements that are directly contributing to the conflict or violation in order to minimize the impact on the other parts of the schedule. However, in order to resolve certain classes of conflicts, multiple tracks not directly associated with the conflict may need to be modified. A strategy that addresses these types of conflicts is discussed next.

The stochastic relayout algorithm generates a new schedule based on adjustments made to existing tracks in the schedule. The algorithm loops through each track in the schedule and stochastically updates any or all of the parameters including start time, duration, antenna, and so on. Each new schedule generated attempts to reduce the number of track conflicts and request violations, thus addressing the issue with single-requirement repair as it is able to find solutions that require modifying multiple tracks that are not directly related to the conflict/violation. Compared to initial layout and basic repair, this strategy was able to reduce the number of conflicts and violations in several test schedules by more than 40 percent.

Conclusions

We have described the DSN scheduling engine component of the service scheduling software (S^3) system, a new scheduling system for NASA's Deep Space Network. The DSE implements a request-driven approach to scheduling, incorporating a sophisticated request specification language, algorithms for generating tracks, resolving conflicts, and repairing request violations, and a distributed architecture to provide high-availability service to a large number of simultaneous users. For more than a year, the DSE with only a test GUI provided the first step of the DSN scheduling process by integrating requirements from all users into a preview schedule. Currently the S^3 system is in full operation, with a browser-based GUI supporting a geographically distributed user base engaged in collaborative peer-to-peer scheduling.

At the present time, the S^3 software is being extended to handle long-range and forecasting functionality. By necessity, there are many similarities between the DSN mid- and long-range planning and scheduling functions. Underlying both is the set of current and future DSN assets, including antennas and equipment, some coming into service and others being decommissioned. Both are based on DSN usage requirements from a varying mission set with a wide range of time-dependent tracking and navigation needs. Both are charged with arriving at an ultimately feasible allocation of DSN resources by balancing user needs and resolving periods of resource contention.

Building on these similarities, the first phase of development of the loading analysis and planning software (LAPS) will make direct use of a number of capabilities already deployed operationally in the midrange S^3 software (see figure 9), including the model of DSN asset availability for antennas and equipment, user and mission types, multispacecraft constellations, and MSPA groupings and their special scheduling rules. Additionally, LAPS will be able to

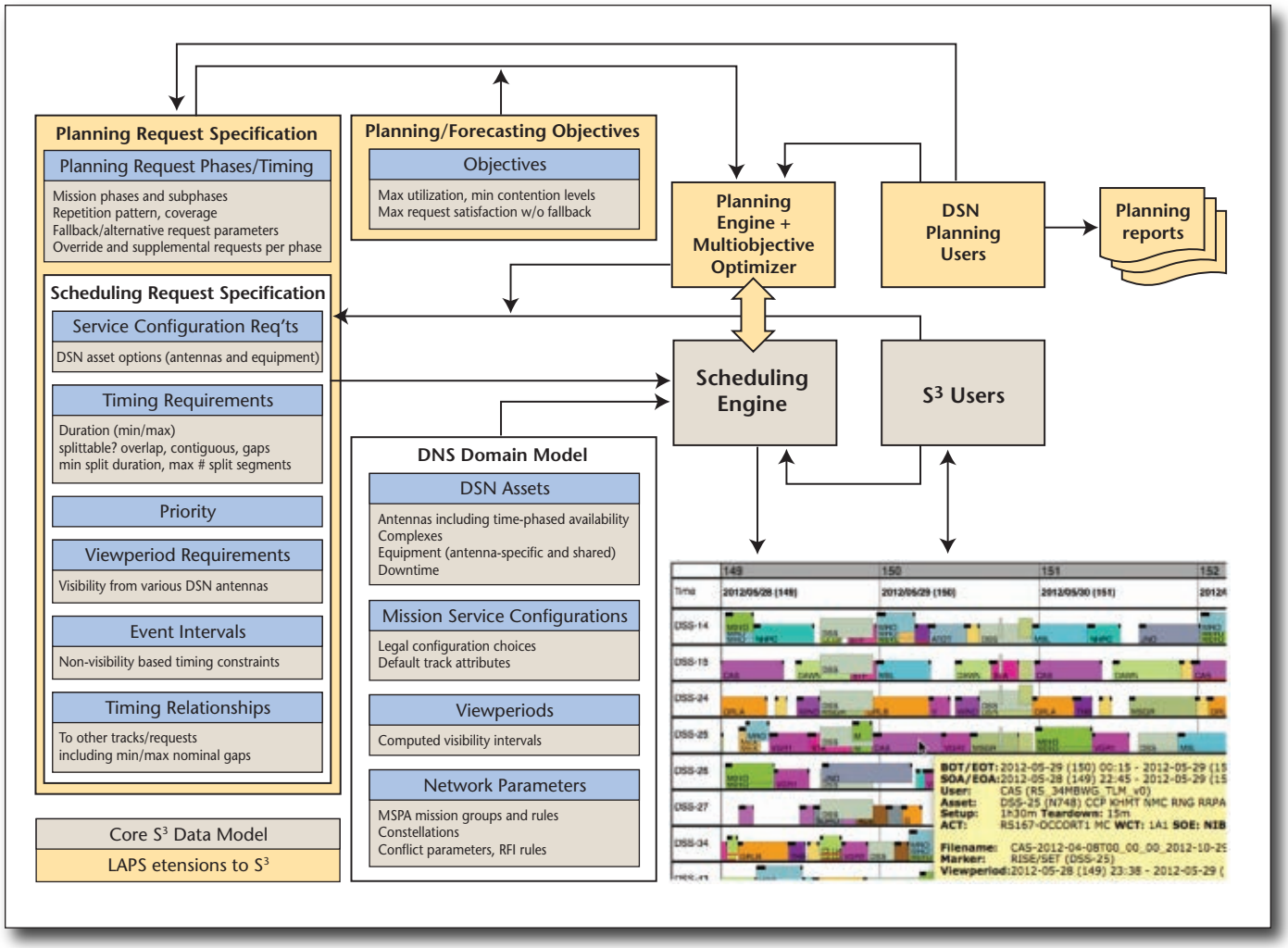


Figure 9. The DSE Data Model and Key Data Flows.

The figure illustrates user interactions with the DSN scheduling engine. The extension of midrange S^3 functionality to support long-range planning and forecasting is highlighted.

invoke the DSE algorithms used in the midrange process, which will allow for fully detailed what-if generation of hypothetical midrange schedule periods in those cases where sufficient detail is available to warrant this level of analysis.

Several other areas are also being addressed with additional capabilities including the following: (1) a planning request representation to allow for more abstract and high-level specification of allocation needs than the scheduling requirement model allows (for example 3x 8hr tracks/week on 34-meter BWG for the 6 months of interplanetary cruise); at the same time, planning requests will be convertible automatically into midrange scheduling requests in order to minimize duplicate data entry and speed up the midrange process; (2) the capability to define and run planning scenarios in an automated way, such as to assess a range of options for down time placement; to evaluate nominal and fallback requirement options for resource contention periods; and to quantify the impact of a mission's alternative launch dates on projected resource loading; and (3) a multiobjective optimization mechanism to automatically generate a portfolio of candidate plans/schedules optimizing the trade-offs among multiple quantitive objectives.

The incorporation of multiobjective optimization (for example, Brown and Johnston [2013]; Johnston [2006]) into LAPS offers a new way to optimize DSN resource allocations, taking into account that there is no single objective that captures all of the disparate goals and objectives that are important. Multiobjective optimization has been employed in a wide variety of problem domains, including scheduling for science missions and generating some requirements inputs to the DSN midrange process (Johnston and Giuliano 2011).

Beyond long-range planning and forecasting,

future work includes extending the scope of S^3 to support near real-time scheduling and cross-network scheduling scheduling capabilities.

Extending the scope of S^3 to support near real-time scheduling, the third phase of the DSN scheduling process, covers the period from execution out to some number of weeks in the future. Extending S^3 to support this phase involves some challenging technical problems of integration with existing systems and support for contingency scheduling (for example, launch slips, unplanned asset down time) as well as operation at the remote DSN complexes; at the same time, bringing the information model of S^3 into the real-time domain will allow for improved decision making considering options that are not now accessible

In addition to the Deep Space Network, NASA also operates two other networks with similar communications and navigation support for certain types of missions: these networks are the Space Network (SN) and Near-Earth Network (NEN). For those users who require services from two or all three of these networks, such integration would be a source of significantly improved efficiency and cost savings. S^3 has the potential to serve as a common scheduling platform in this regard. It is interesting to note that nowhere on the S^3 scheduling request editor main UI is there any indication that the user is working with the DSN; this is apparent only when drilling down into the detailed visibility intervals and service definitions.

Acknowledgments

The research described in this article was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. The authors gratefully acknowledge the support and involvement of the DSN scheduling user community over the course of this work.

References

Bell, C. 1993. Scheduling Deep Space Network Data Transmissions: A Lagrangian Relaxation Approach. In *Applications of Artificial Intelligence 1993: Knowledge-Based Systems in Aerospace and Industry*. SPIE 1963. Bellingham, WA: Society of Photo-Optical Instrumentation Engineers.

Biefeld, E., and Cooper, L. 1991. Bottleneck Identification Using Process Chronologies. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence,* 218–224. San Francisco: Morgan Kaufmann Publishers.

Brown, M., and Johnston, M. D. 2013. Experiments with a Parallel Multiobjective Evolutionary Algorithm for Scheduling. Paper presented at the 5th International Workshop on Planning and Scheduling for Space, Baltimore, MD, 22–25 October.

Carruth, J.; Johnston, M. D.; Coffman, A.; Wallace, M.; Arroyo, B.; and Malhotra, S. 2010. A Collaborative Scheduling Environment for NASA's Deep Space Network. Paper presented at the 10th International Conference on Space Operations (SpaceOps 2010), Huntsville, AL, 25–30 Apr.

Chien, S.; Johnston, M. D.; Frank, J.; Giuliano, M.; Kavelaars, A.; Lenzen, C.; and Policella, N. 2012. A Generalized Timeline Representation, Services, And Interface For Automating Space Mission Operations. Paper presented at the 12th International Conference on Space Operations (SpaceOps 2012), Stockholm, Sweden, 11–15 June.

Chien, S.; Rabideau, G.; Knight, R.; Sherwood, R.; Engelhardt, B.; Mutz, D.; Estlin, T.; Smith, B.; Fisher, F.; Barrett, T.; Stebbins, G.; Tran, D. 2000. ASPEN — Automating Space Mission Operations Using Automated Planning and Scheduling. Paper presented at the 1st International Conference on Space Operations (SpaceOps 2000), Toulouse, France, 19-23 June.

Chien, S. A.; Hill, R. W. J.; Govindjee, A.; Wang, X.; Estlin, T.; Griesel, M. A.; Lam, R.; Fayyad, K. V. 1997. A Hierarchical Architecture for Resource Allocation, Plan Execution, and Revision for Operation of a Network of Communications Antennas. In *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*. Piscataway, NJ: Institute of Electrical and Electronics Engineers. dx.doi.org/10.1109/ROBOT.1997.606798

Clement, B. J., and Johnston, M. D. 2005. The Deep Space Network Scheduling Problem. In *Proceedings of the 17th Conference on Innovative Applications of Artificial Intelligence,* 1514–1520. Menlo Park, CA: AAAI Press.

Fisher, F.; Chien, S.; Paal, L.; Law, E.; Golshan, N.; and Stockett, M. 1998. An Automated Deep Space Communications Station. In *Proceedings of the 1998 IEEE Aerospace Conference,* volume 3. Piscataway, NJ: Institute of Electrical and Electronics Engineers. dx.doi.org/10.1109/AERO.1998.685791

Guillaume, A.; Lee, S.; Wang, Y.-F.; Zheng, H.; Hovden, R.; Chau, S.; Tung, Y.-W.; Terile, R. J. 2007. Deep Space Network Scheduling Using Evolutionary Computational Methods, 1–6. In *Proceedings of the 2007 IEEE Aerospace Conference*. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Imbriale, W. A. 2003. Large Antennas of the Deep Space Network. New York: Wiley. dx.doi.org/10.1002/0471728497

Johnston, M. D. 2006. Multi-Objective Scheduling for NASA's Deep Space Network Array. Paper presented at the 5th International Workshop on Planning and Scheduling for Space (IWPSS-06), Baltimore, MD, 22–25 October.

Johnston, M. D., and Clement, B. J. 2005. Automating Deep Space Network Scheduling and Conflict Resolution. Paper presented at the 2005 International Symposium on Artificial Intelligence, Robotics, and Automation for Space, Munich, Germany 5–9 September.

Johnston, M. D., and Giuliano, M. 2011. Multi-Objective Scheduling for the Cluster II Constellation. Paper presented at the 7th International Workshop on Planning and Scheduling for Space 2011. IWPSS-2011 Darmstadt, Germany, June 8–10.

Johnston, M. D.; Tran, D.; Arroyo, B.; and Page, C. 2009. Request-Driven Scheduling for NASA's Deep Space Network. Paper presented at the 6th International Workshop on Planning and Scheduling for Space (IWPSS-06), Pasadena, CA, 19–21 July.

Johnston, M. D.; Tran, D.; Arroyo, B.; Call, J.; and Mercado, M. 2010. Request-Driven Schedule Automation for the Deep Space Network. Paper presented at the 10th International Conference on Space Operations (SpaceOps 2010), Huntsville, AL, 25–30 Apr.

Kan, E. J.; Rosas, J.; and Vu, Q. 1996. Operations Mission Planner — 26M User Guide Modified 1.0. JPL Technical Doc-

ument D-10092. Pasadena, CA: Jet Propulsion Laboratory, California Institute of Technology.

Lacey, N., and Morris, D. G. 2002. JPL RAPSO Long-Range Forecasting. Paper presented at the 12th AAS/AIAA Space Flight Mechanics Meeting, San Antonio, Texas, 27–30 January.

Loyola, S. J. 1993. PC4CAST: A Tool for DSN Load Forecasting and Capacity Planning. NASA Technical Report 19940009911. Pasadena, CA: Jet Propulsion Laboratory, California Institute of Technology.

Werntz, D.; Loyola, S.; and Zendejas, S. 1993. FASTER — A Tool for DSN Forecasting and Scheduling, 230–235. Paper presented at the 9th American Institute of Aeronautics and Astronautics (AIAA) Computing in Aerospace Conference, October 19–21, San Diego, CA.

Mark D. Johnston is a principal scientist in the Planning and Execution Systems section at the Jet Propulsion Laboratory, California Institute of Technology. His research interests include reactive planning and scheduling, multiobjective optimization, and evolutionary algorithms. His background is in physics, with a bachelor's degree from Princeton and Ph. D. from MIT. He has worked on a wide range of planning and scheduling technologies in both space and commercial domains, and is the originator of the Spike scheduling system, used for Hubble Space Telescope as well as a number of other space- and ground-based observatories. He currently leads the JPL Deep Space Network Service Management development team, and also the SCaN network integration project increment 1 team.

Daniel Tran is a senior member of the technical staff in the Artificial Intelligence Group at the Jet Propulsion Laboratory, California Institute of Technology. Tran received a B.S. in computer engineering from the University of Washington and M.S. in computer science from the University of Southern California. He was the flight software lead for the autonomous sciencecraft experiment, cowinner in the 2005 NASA Software of the Year competition. He was also software lead for the service scheduling software (SSS), the primary system used for midrange scheduling of NASA's Deep Space Network.

Belinda Arroyo manages the scheduling process for NASA's Deep Space Network. She has more than 20 years experience at JPL in mission scheduling, mission operations, and deep space navigation, and is the recipient of a number of awards in these areas. She has overseen the deployment of the new scheduling software described in this article, and the consequent adaptation of scheduling processes.

Sugi Sorensen is a senior systems engineer at the Jet Propulsion Laboratory. He currently provides systems engineering support to the Deep Space Network project in the scheduling process realm. He also serves as the operations engineer for the service scheduling software (SSS) system. His previous work at JPL includes systems engineering support for the Planetary Data System and he served as a group supervisor and product manager in the Advanced Concepts Engineering group where he specialized in human computer interface design and web-based information systems.

Peter Tay is a user support specialist at the Jet Propulsion Laboratory, using the service scheduling software (SSS) for the Deep Space Network midrange scheduling process. He is responsible for reducing hard contentions to a manageable level for the DSN scheduling community, before they begin their negotiation process. Tay has previously supported various missions as flight project schedule engineer for 16 years, including the Mars missions, Cassini, Voyagers 1 and 2, Stardust, and Genesis, as well for several non-NASA missions.

Butch Carruth is founder and president of Innovative Productivity Solutions, Inc., specializing in software-based task automation. His work in the space industry began in 2000 as architect of the science planning software for the Cassini mission to Saturn. He and his team have been working with the Jet Propulsion Laboratory on software solutions for the Deep Space Network since 2006. He continues to support a variety of deep space and low Earth orbit missions in a software development and data management capacity.

Adam Coffman is a senior developer at Innovative Productivity Solutions where he started in 2002 after graduating with a computer science degree from Trinity University in 2001. He started on the Cassini mission in 2001 and as been focused on collaborative scheduling solutions for the Deep Space Network since 2006.

Mike Wallace is a software engineer at IPS, Inc. He works on database systems, website design, and data analysis. He has worked on multiple NASA software projects including visualization tools for mission planning and mission design, telemetry analysis and storage, and antenna scheduling.

Leveraging Multiple Artificial Intelligence Techniques to Improve the Responsiveness in Operations Planning: ASPEN for Orbital Express

Russell Knight, Caroline Chouinard, Grailing Jones, Daniel Tran

■ *The challenging timeline for DARPA's Orbital Express mission demanded a flexible, responsive, and (above all) safe approach to mission planning. Mission planning for space is challenging because of the mixture of goals and constraints. Every space mission tries to squeeze all of the capacity possible out of the spacecraft. For Orbital Express, this means performing as many experiments as possible, while still keeping the spacecraft safe. Keeping the spacecraft safe can be very challenging because we need to maintain the correct thermal environment (or batteries might freeze), we need to avoid pointing cameras and sensitive sensors at the sun, we need to keep the spacecraft batteries charged, and we need to keep the two spacecraft from colliding ... made more difficult as only one of the spacecraft had thrusters.*

Because the mission was a technology demonstration, pertinent planning information was learned during actual mission execution. For example, we didn't know for certain how long it would take to transfer propellant from one spacecraft to the other, although this was a primary mission goal. The only way to find out was to perform the task and monitor how long it actually took. This information led to amendments to procedures, which led to changes in the mission plan. In general, we used the ASPEN planner scheduler to generate and validate the mission plans. ASPEN is a planning system that allows us to enter all of the spacecraft constraints, the resources, the communications windows, and our objectives. ASPEN then could automatically plan our day. We enhanced ASPEN to enable it to reason about uncertainty. We also developed a model generator that would read the text of a procedure and translate it into an ASPEN model. Note that a model is the input to ASPEN that describes constraints, resources, and activities. These technologies had a significant impact on the success of the Orbital Express mission. Finally, we formulated a technique for converting procedural information to declarative information by transforming procedures into models of hierarchical task networks (HTNs). The impact of this effort on the mission was a significant reduction in (1) the execution time of the mission, (2) the daily staff required to produce plans, and (3) planning errors. Not a single misconfigured command was sent during operations.

ISSN 0738-4602

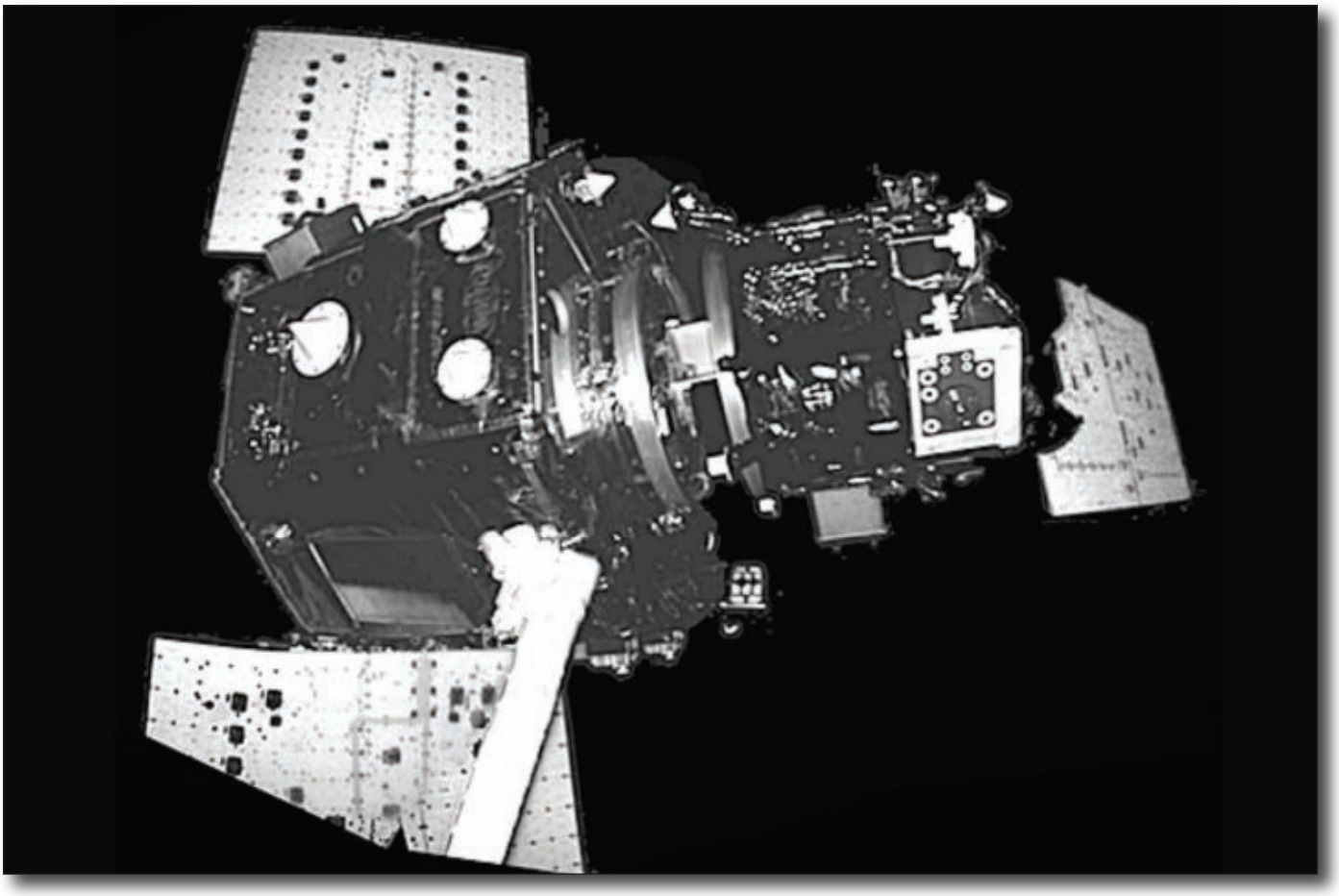


Figure 1. The ASTRO and NextSat Spacecraft, on Orbit Without the Separation Ring.

It is often the case that new technology for space needs to be taken out of the lab and proven in space. The primary goal of a technology mission is to prove the capabilities of newly developed space technology. Most technology missions have the challenge of discovering the limits and capabilities of new systems, and the Defense Advanced Research Projects Agency's (DARPA) Orbital Express (OE) mission was no exception. Orbital Express launched March 8, 2007, and decommissioned on July 22, 2007. The Orbital Express mission demonstrated on-orbit servicing of spacecraft, including rendezvous, transfer of battery and CPU modules, and transfer of propellant, the actual duration of each being approximated but not known to high enough fidelity to commit to a communications and operations plan.

This introduced challenges because pertinent information needed for planning was not available until the various technology experiments were performed, but to perform these experiments we needed to submit plans to reserve communications resources, configure execution scripts, and monitor execution. We note that it is often the case that bounds of performance are known a priori. Our automated planning system leveraged this information to help address the inherent uncertainty in experiment planning.

The Orbital Express planning cycle consisted of a long-term plan (planned four weeks in advance) and a short-term plan (planned the day before operations). The long-term plan was produced before any of the experimental/unknown information was learned. The role of the long-term plan was to ensure that enough resources are reserved ahead of time. Planning at this point required accommodation of the bounds of possible execution. Traditionally, this level of planning is relatively conservative. For example, if we thought it likely that a propellant transfer would require 20 minutes, but could take up to an hour, we would plan for the whole hour. We also couldn't count on getting all of the communications passes we asked for, so we would plan for 20 percent of them to be dropped, and thus ask for more than we needed.

The short-term plan was produced immediately before execution, and some of the experimental or

Figure 2. The Ejected Separation Ring.

NextSat (top) is at the end of the robotic arm (upper left).

unknown information is known and should be integrated. Also, we knew with certainty which communications passes we would be granted. This allowed us to free resources and helped us to reduce cost or reduce the risk of other missions using the shared resources.

Mission planning had many challenges. First, there was a large degree of uncertainty in the execution of our tasks, such as, pumping propellant; second, the procedures for operations were changing within hours of the time that we had to deliver our plans for execution; third, we had to predict how the on-board execution system would behave so that we could accommodate its behavior, even though we were not planning for it explicitly; and fourth, the problem of building a plan even for single day was monumental: hundreds of constraints needed to be checked for thousands of actions, and all needed to be coordinated with procedures that were being developed: building a plan by hand was simply infeasible, regardless of staffing.

Technologies that address each of these challenges were leveraged in developing the Orbital Express ground-planning system are schema-level uncertainty reasoning (for long-term planning), procedure parsing for model generation (for short-term planning), procedural to declarative model translation, and, most importantly, automated planning and scheduling.

DARPA's Orbital Express mission demonstrated on-orbit servicing of spacecraft. Servicing spacecraft has a great potential to increase the lifespan of these exceedingly expensive assets, but the complexity involved in servicing a spacecraft on orbit had been overwhelming. Consider that all spacecraft in low Earth orbit will fall to Earth unless they expend propellant to stay in orbit. If we could pump more propellant into these, we would be giving them new life. Add to this the potential of replacing modules, such as batteries or central processing units (CPUs), then the value of on-orbit servicing becomes clear. Two spacecraft were flown: Boeing's Autonomous Space Transport Robotic Operations (ASTRO) spacecraft (see figure 1), whose role was that of a doctor. ASTRO had a robotic arm, replacement modules (battery and CPU), a propellant pumping system, and a capture device (used to grasp other spacecraft and lock in the propellant pumping mechanism). Ball Aerospace's Next Generation Serviceable Satellite (NextSat) spacecraft (see figures 1 and 2) had the role of a patient. NextSat could receive propellant and modules, but it couldn't maneuver because it had no thrusters. It was up to ASTRO to perform all of the maneuvering. Experiments included rendezvous and capture, fluid propellant transfer, and on-orbit repair.

The mission planning team was divided into two units, the rendezvous planners who concerned themselves primarily with computing the locations and visibilities of the spacecraft, and the scenario resource planners (SRPs) who were concerned with assignment of communications windows, monitoring of resources, and sending commands to the ASTRO spacecraft. The SRP position was staffed by Jet Propulsion Laboratory (JPL) personnel who used the Activity Scheduling and Planning Environment (ASPEN) planner-scheduler.

We discuss the Orbital Express domain in the context of ASPEN, the technologies added to the planner to accommodate the mission objectives, and the ground operations of long-range and daily planning for the mission.

Mission Operations Planning

The OE mission domain required fast turnaround of heterogeneous, dynamic plans. Every day was a different scenario, where communication passes could be lost within hours of the daily planning delivery deadline. Procedures could change within hours of the delivery deadline because the impacts of on-orbit experiments on spacecraft resources (energy and memory) were to a significant extent unknown. Limited available communications existed using primarily the high-bandwidth ground-based Air Force Satellite Control Network (AFSCN) sites, while the relatively low-bandwidth GEO-Synchronous spaceborne tracking and data relay satellite system (TDRSS) communications could potentially vary by the hour. The main difference between AFSCN and TDRSS is

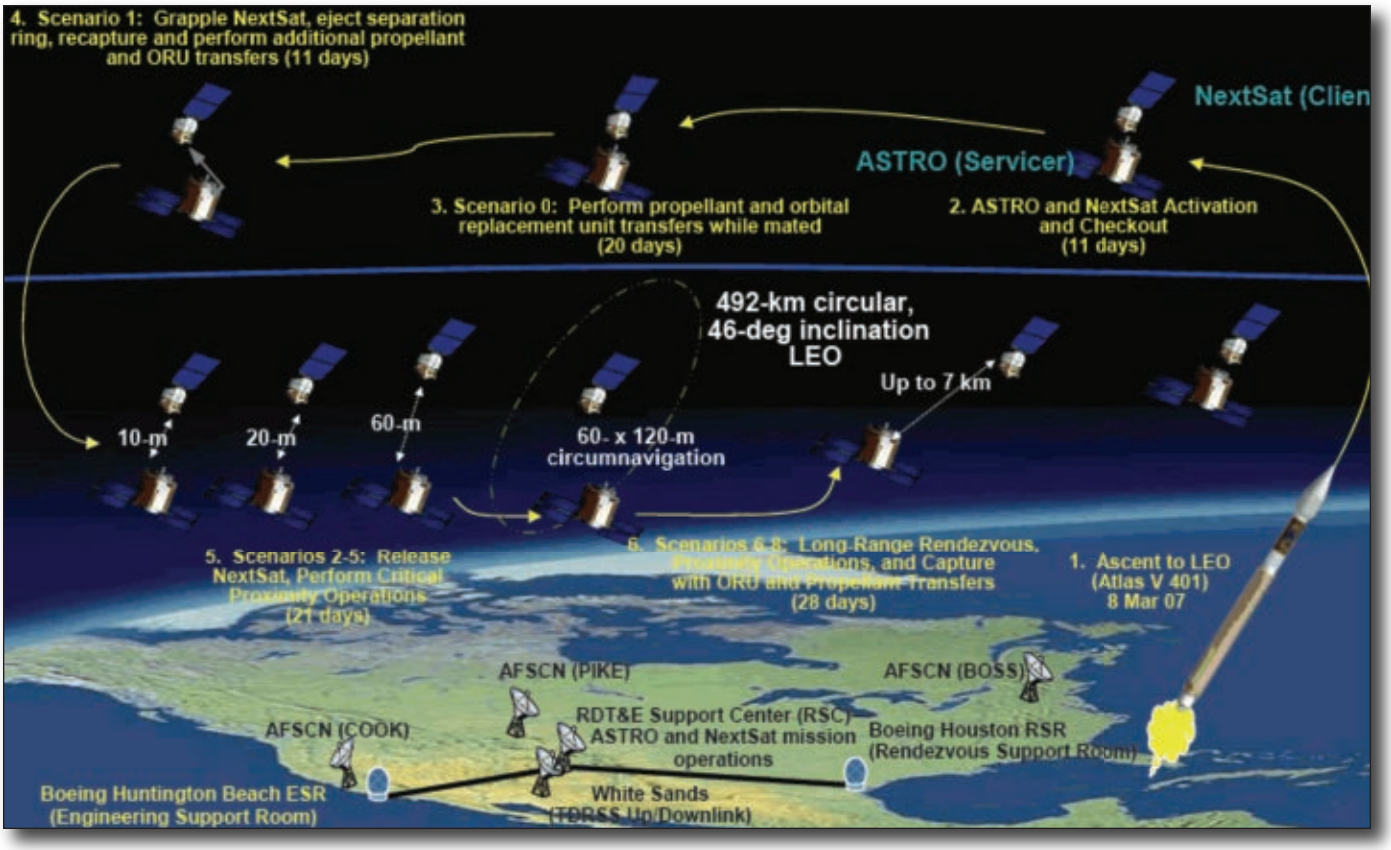


Figure 3. The Orbital Express Scenarios: Increasing Autonomy and Complexity.

that AFSCN sites are ground-based and are only in view for about 5 minutes (when in low Earth orbit, as ASTRO and NextSat were), but TDRSS is satellite-based and is pretty much always in view. The challenge is that these resources are shared across many missions, and each mission has to reserve time in advance. But events can cause resources to become available or to drop out, so it is often the case that we need to be able to respond to these changes quickly.

A scenario (see figure 3) typically consisted of a series of procedures, each of which was written by the operations personnel in Microsoft Word table format. Each procedure listed its steps and associated durations and represented the need for a contact and the type of contact desired or required. Several procedures had other embedded procedures and some spanned more than one day. As an example, the unmated scenario required an initial setup procedure, then the unmated procedure would be kicked off; the de-mate, hold, and re-mate would execute, and then a postrendezvous and capture transfer procedure would be planned. See figure 4 for images of the unmated scenario midexecution in the de-mated configuration and in the final stages of berthing to the mated state.

The schedule of each scenario was dependent on what had been accomplished to date, as the goal of each scenario was to become increasingly more autonomous. The planning schedule was also dependent on the state of the flight system, the amount of preparation time needed before execution, and resources available on future dates. Calendar planning was done by a combination of inputs from flight directors, mission managers, project management, and DARPA.

Procedures were delivered to the SRP and copied to Excel. An ASPEN model-generation script was then used to create ASPEN Modeling Language (AML) representations of the procedures. Once the AML model existed for a procedure, the ASPEN tool read the AML description of the procedure and could be used to add any number of different procedures in a plan required to make up the scenario. See the data flow diagram and the final daily plan in figure 5.

Roles of Mission Planning

Mission planning had two primary roles for Orbital Express: (1) evaluate scenarios for feasibility early in the design of the mission, and (2) provide responsive communications and commanding planning and scheduling during the mission. To serve both roles,

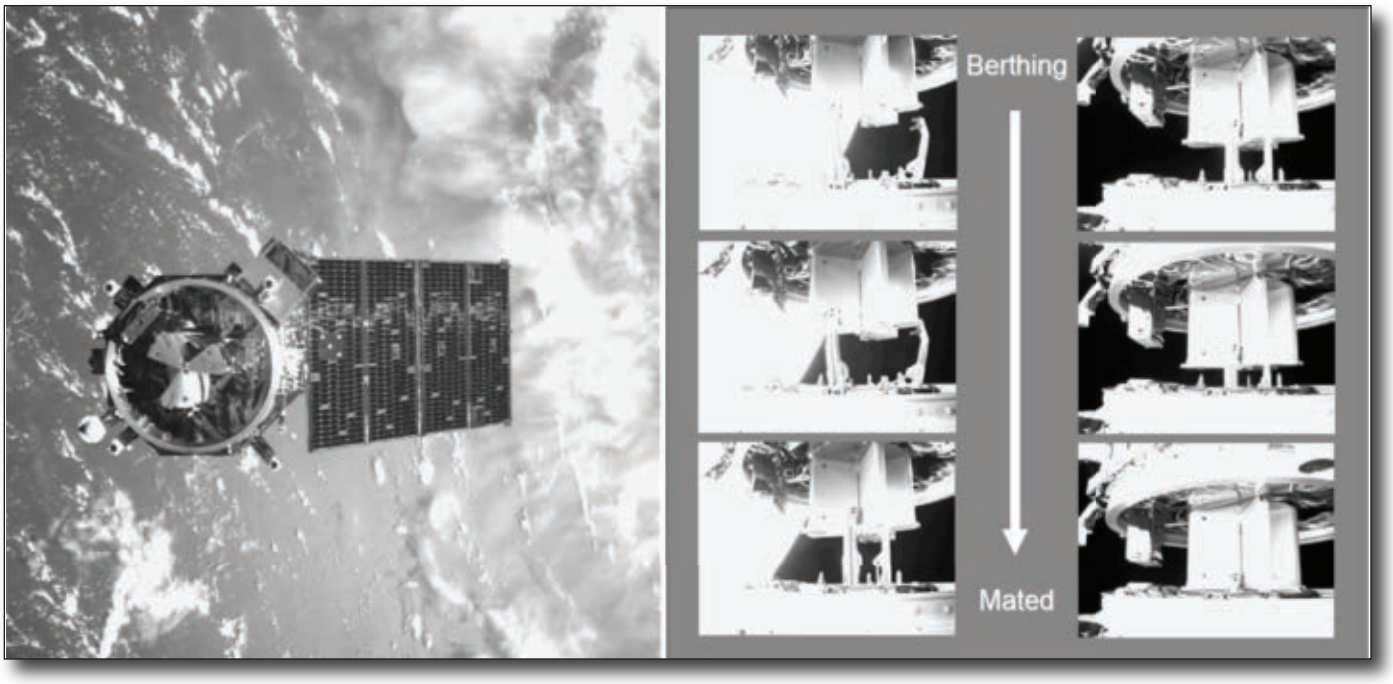


Figure 4. A Demate/Mate Scenario.

NextSat is 14m away during a departure, then progressive side view configurations of the "Berthing" to "Mated" states are shown.

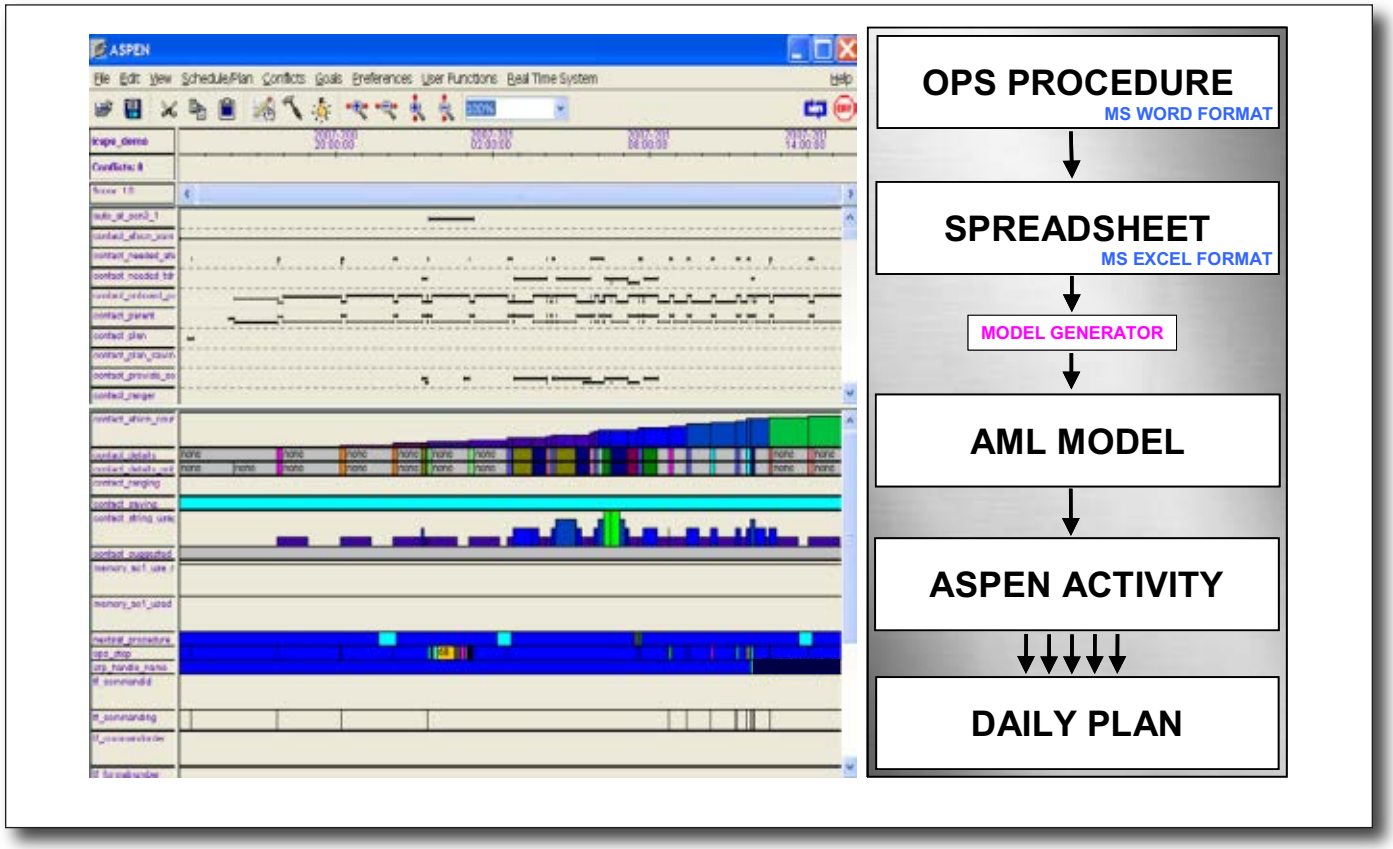


Figure 5. The ASPEN Planner-Scheduler Displays a Daily Plan.

we modeled the mission scenarios using the ASPEN (Rabideau et al. 1999) planning system. OE required evaluation of many alternatives, so ASPEN was modified to accommodate reasoning about schema-level uncertainty. Rehearsals for operations indicated that the SRP needed to be very responsive to changes in the procedures. To accommodate this, we implemented a system for reading the procedures and interpreting these into ASPEN models.

Technologies

The technologies we leveraged were (1) schema-level uncertainty reasoning, (2) procedure parsing for model generation, (3) procedural to declarative model translation, and (4) automated planning and scheduling. Schema-level uncertainty reasoning has at its core the basic assumption that certain variables are uncertain but not independent. Once any are known, then the others become known. This is important where a variable is uncertain for an action and many actions of the same schema (or type) exist in the plan. For example, the number of retries to purge the pump lines were unknown (but bounded), and each attempt required a subplan. Once we knew the correct number of attempts required for a purge, it would likely be the same for all subsequent purges. To accommodate changing scenario procedures, we ingested the procedures into a tabular format in temporal order, and used a simple natural language parser to read each step and derive the impact of that step on memory, power, and communications. We then produced an ASPEN model based on this analysis. That model was tested and further changed by hand, if necessary, to reflect the actual procedure. This resulted in a great savings in time used for modeling procedures.

Schema-Level Uncertainty Reasoning

To accommodate schema-level uncertainty reasoning in ASPEN, we modified the ASPEN Modeling Language to include a new reserved word — uncertain. Any parameter of any activity type that was unknown (but bounded) would be labeled using this reserved word, such as,

uncertain int retries = [1, 2]

or

uncertain string mode = (idle, transmitting, off).

Then, when an instance of ASPEN was started with uncertain variables, the cross-product of the instantiations of uncertain variables was used to produce unique instances of plans. Each of these instances is called an alternative. Note that this is the cross-product of the schema-level instantiations, not the actual activity-level instantiations. If we take our previous example, we would instantiate six alternatives:

retries = 1, mode = “idle”

retries = 1, mode = “transmitting”

retries = 1, mode = “off”

retries = 2, mode = “idle”

retries = 2, mode = “transmitting”

retries = 2, mode = “off”

Now, every activity in each alternative would have the same value, so it wouldn’t matter how many activities we had. This differs greatly from activity-level uncertainty. In this case, we would need to generate an alternative for each possible activity assignment. This means that we would have exponentially many alternatives with increasing activities. Since the uncertain parameters are those that we expect to learn (and to not vary), then we can expect that if a parameter has a value earlier in the day, it will have the same value later in the day.

Also, operations staff was loathe to trust a more analytical and compressed form of uncertainty reasoning. It was a very compelling case to see all possible executions, and when they needed to justify a certain resource allocation they found it simple and intuitive to use the set of alternatives.

To perform planning, we plan each alternative as if it was a separate schedule, and then perform a merge of the schedules, resulting in what operations people consider to be “odd” schedules, where we might ask for resource allocations that are impossible for a single spacecraft but still must be accommodated if we are to accommodate all possible alternatives. If we are not granted an allocation, we can go to each alternative and either try to replan it or simply carry it as a risk.

In practice, uncertain labels were used judiciously, not only to reduce the size of the set of problems to solve, but also to keep the solutions presented within the space of what humans could inspect and certify. The largest cross-product of schemas produced at most 32 separate plans.

Procedure Parsing for Model Generation

To accommodate late changes in procedures we implemented software that read procedures and produced ASPEN models. At first, this seemed like a daunting problem: we are in essence reading English text for content and producing a declarative activity/time line–based model of the procedure. One key observation we made is that the language of procedures is nearly as strict as a programming language, so we did not need to produce a parser capable of complete natural language processing; we just needed to accommodate stylistic differences between authors. Of course, some free-form text does appear in the procedures, and the model needed to be annotated such that the ASPEN model parser would complain in a meaningful way and the human modeler would address the text that was not understood.

This highly circumscribed form of natural language arose from the fact that these procedures were to interleave human actions on the ground and

machine actions in space. This is in stark contrast to other procedures (such as International Space Station [ISS] procedures) that might leave much to the interpretation of the reader and require training to be able to understand and perform, although currently efforts are under way to make ISS procedures more formally structured (Kortenkamp, Bonasso, and Schreckenghost 2008).

The procedures consisted of an introduction of human readable text, followed by a table of steps. They were authored using Microsoft Word. We found that most of the information needed to generate the procedure model was available in the table, so we would copy and paste the table into a Microsoft Excel document. Our parser was written in Visual Basic and embedded in the Microsoft Excel document.

Each step of the procedure had a number, the position or role responsible for carrying out the step, the action that was taking place, the response or verification to the action, and the expected duration. By parsing the action, we could determine whether the step included loops, if statements, or commands.

Loops in the procedures were accommodated using recursive decompositions. In ASPEN, it is often convenient to model activities and subactivities in trees known as hierarchical task networks. This representation is handy, but does not accommodate dynamic structures in the hierarchy. But it does allow for disjunctions, for example, an activity heater_parent can decompose into either a heater_child or a dummy activity. If we allow loops in the hierarchy, we can represent dynamic structures. The problem introduced by this is that the hierarchy appears to be infinitely deep. Therefore, we need to ensure that there are termination criteria; that is, at some point the loop breaks out to a subbranch that has no loops.

If statements were modeled using disjunctive decompositions. Both loops and ifs were candidates for uncertain variables.

The table also had commands that were to be sent to the spacecraft at execution time. Some of these commands were simple in that no further information was needed. In this case, the command activity was included as part of a decomposition. But, some of the commands required information to be input or computed. In this case, a human modeler needed to decide on the information source. To keep this from accidentally generating a working model, we would assign a known nonexistent variable the string value of the text describing the command argument. To ensure that command arguments and mnemonics were correct, we produced an ASPEN model from the command dictionary stored in a Microsoft SQL database. This was a piece of SQL code written by Boeing personnel. This included the legal bounds for each argument.

If any procedure had poorly formed commands, the ASPEN parser would catch them, and the procedure would be corrected. This was a relatively free value-added effect that resulted in the correction of many procedures.

Procedural to Declarative Model Translation

We have hinted at the necessity of converting procedural modeling information into declarative models. ASPEN is by design a declarative system. The procedures that were parsed were by nature procedural … meaning that each consists of a series of steps that include blocks, decision points (if statements), and loops. We not only needed to model the ground procedures but also had to model the on-board sequencer's (Timeliner) behavior. These were a collection of scripts that were executed for each command. The necessity of modeling the on-board execution comes from the requirement to model power use and to accommodate communication windows. These scripts were translated into ASPEN Modeling Language similarly to the previously mentioned procedures.

1. Each step in time is modeled as an individual activity, with the variables of interest being parameters in the activity.

2. Each series of steps was modeled as a single block, with a parameter network being constructed that represented the execution of the series of steps, as one might model a program execution in several rows in Excel.

3. Each if statement was modeled as a hierarchical activity with a disjunctive decomposition (a this-or-that decomposition). Constraints were imposed that forced the selection of the correct decomposition according the expression being evaluated by the if statement.

4. Each loop was modeled as a hierarchical activity with a disjunctive decomposition that included a recursion (that is, an activity lower in the instantiated hierarchy could be of the same type or schema as an activity higher in the hierarchy.) Note that termination criteria should default to terminating or the decomposition would expand endlessly on instantiation.

Ad Hoc Adjustments

Of course, the models needed to be maintained. As the mission progressed, unknown variables were adjusted to best estimate reality; for example, a hydrazine propellant transfer became more predictable after several were successfully demonstrated. Any model representing a procedure could need updating over time. There were cases in which the values simply changed; for example, the rate of a fuel transfer needed updating. However, there were also cases in which steps in the procedure needed to be removed or, more difficult from a planning perspective, added. To remove a step, the duration of the step could be set very low, or in the worst case, the procedure model could simply be regenerated. To

add steps, we almost always simply regenerated the model from the procedure using the translator.

Automated Planning

Underlying all was the ASPEN planner-scheduler. No adaptation-specific code was used for Orbital Express … all of the adaptation used the out-of-the-box capabilities of ASPEN. To plan any given day the following steps were followed:

1. Load all of the known opportunities (AFSCN visibilities, TDRSS visibilities, and procedures that have already been preplanned by hand or from previous runs of ASPEN).

2. Instantiate as many alternative plans as necessary to accommodate the uncertain parameters. In practice, long-term plans had several alternatives but short-term plans had only one.

3. Schedule all activities using an earliest-deadline-first ordering, also known as forward dispatch scheduling. This results in realigning activities to the known availabilities and initial expansion into the supporting subactivities.

4. Iteratively repair the plan to address any plan flaws (for example, unexpanded decompositions, open temporal associations between activities) and constraint violations (such as resource oversubscription, state violations, and timing violations).

Each of these steps was automatically performed by the ASPEN system under the direction of the SRP.

Long-Range Planning

The planning process for any given day began weeks in advance. A plan was built from knowledge of the existing contacts available and an ASPEN-generated and edited model of what the procedure was to do and how the contacts should lay out across time (figure 6)

The AFSCN contacts were reserved up to a limit and occasionally with elevated priorities specifically for the unmated scenarios. TDRSS support was originally also scheduled in the long-range planning time frame for all scenarios; however, cost constraints and changes to the plan in the short term dictated the need for a policy change.

It was determined more efficient to schedule TDRSS at the daily planning time, except in the case of unmated scenarios, where the timing and the more definite guarantee of contacts was crucial.

Although the essential replanning generally occurred at the daily planning time, variations on the long-range planning occurred from several factors. First, our launch delay created the need to replan all existing long-range plans to shift both AFSCN and TDRSS requests. Second, changes to models occurred often during the long-range process, due to many factors, including updated knowledge of timing, procedure step removals and additions, and general modifications to procedure step times or requirements. Third, occasionally, maintenance requirements or site operating hours were learned postdelivery of the long-range planning products and a replan was necessary. Finally, other factors that required replanning the long-range products were often late enough in the plan time line that a new "midrange" plan was created. This usually was done a few days outside of the daily planning. Figure 6 depicts the planning flow.

Daily Planning

In the morning of daily planning, the SRP would receive the list of contacts lost to other spacecraft and any suggested additions to replace these losses, and he or she would also receive the most up-to-date list of TDRSS availabilities. The contact losses would need to be evaluated against the procedure objectives of the day to determine whether they could still be met. The ASPEN model of the procedure could be adjusted as needed to reflect any operations updates, and the ASPEN activity could be moved around throughout the day to accommodate the contact requirements.

In the nominal case, the planning process would call for the use of the long-range plan and simply update necessary timing information to create the daily plan. However, daily planning was based on many variable factors culminating in a need for both simple updating of the plan and completely replanning the long-range plan: (1) The visibilities of contacts with the position of the spacecraft drifts slightly per day and must be updated in the short term to make most efficient use of the AFSCN communication times. Even one minute of contact coverage loss was, at times, considered valuable. (2) The daily deconfliction process can mean a loss of several contacts based on any number of reasons (site-specific issues, other satellite conflicts). Losses may require a shift of the procedure to perform the requested objectives. Also, losses are often accompanied by gains, and replanning can be based on such new additions to the plan. (3) Scoping of the day's long-range plan may change due to both anomalies and new direction from operations. Updating the existing plan at the daily planning time was often required for previously unknown amounts of needed coverage or for real-time failures of contacts pushing into the next day. (4) TDRSS support was originally requested in advance for all long-range planning, but as cost became an issue for unused contacts, the requests for TDRSS became part of the daily planning process. This was a major addition to the update of the long-range plan. (5) Dealing with the sometimes unpredictable conditions of space and limited mission time, a number of unforeseen events could cause the need to update the long-range plan.

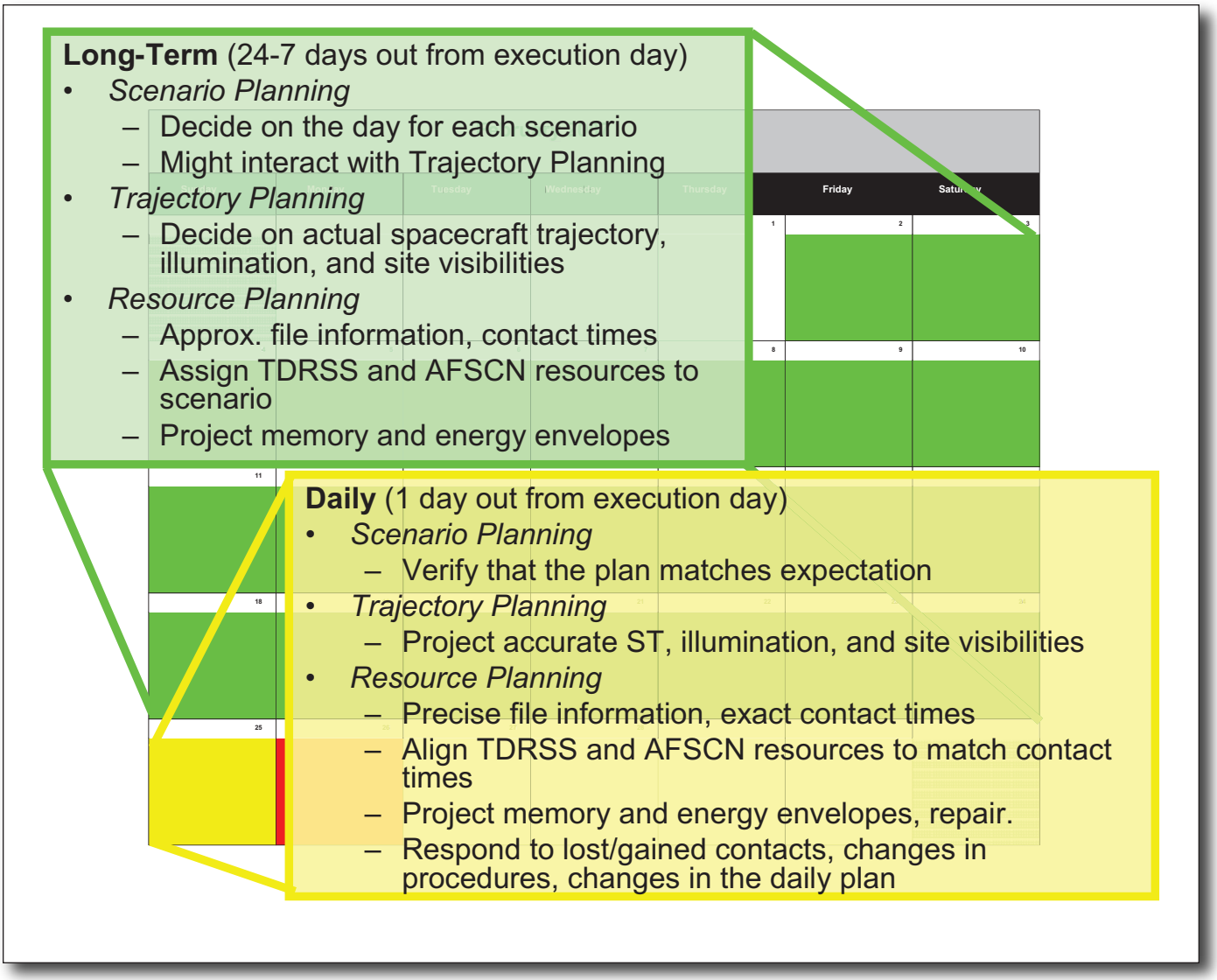


Figure 6. The Planning Flow.

Long Range and Daily Planning.

Impact

We were able to produce several alternatives for long-term planning so that enough communications resources were available at the time of execution. (For each alternative, we needed enough resources to handle all communications. Each alternative was based on differing execution paths.) We also were able to deliver operations plans daily, even in the face of changing procedures and changing resource availability. Together this contributed to the success of the mission.

The overall affect of using ASPEN has been approximated by the flight director as a 26 percent reduction in the execution time of the mission, a 50 percent reduction in the daily staff required to produce plans, and a 35 percent reduction in planning errors. Note that by planning error, we mean errors made by staff in terms of what plans should be executed that day, not errors in the plans themselves. This reduction was due to the high visibility of each plan and ready inspectability by expert staff. In fact, not a single misconfigured command was sent during operations.

All of these savings resulted in an estimated overall cost reduction of $10.4 million, mostly due to the reduction in execution time. Keeping in mind that the total cost of development and operations for the automated ground planning system was less than $1 million, this becomes a clear winner. Note that this does not include any economies of scale (which are normally associated with claims of automation reducing cost) as each major experiment was performed only once.

Lessons Learned

With a 100 percent mission criteria success rate, the Orbital Express project proved that spacecraft servicing is a reality for space operations. The goals for the JPL ASPEN team were to model the procedures and constraints of the mission and plan the long-term and daily operations. Using ASPEN and AML for Orbital Express modeling and planning, the planning team was able to represent mission constraints and procedures. The planning tool was flexible and adaptable to changing parameters.

In the long-term plan time frame, the plan for the execution day often changed or had several alternatives in one day (the nominal plan versus a backup plan). ASPEN's internal activity structure and repair algorithm allowed procedures to be shifted easily from one contact to another and to be deleted and replaced by new procedures without major reworking of the plan. Daily planning was adaptable to changes in which procedures and their associated ASPEN models were updated by operations the day of planning. The autogeneration of models allowed the planning team to share new procedure information and process it quickly for use on orbit. It also allowed the decision of using a procedure on a given day to be made at the very last minute without affecting the mission schedule. Models could be generated during the daily planning process and used the same day to plan that day's contacts.

Originally, NextSat contacts were not going to be scheduled using ASPEN; however, the simplicity of adding objectives to contacts with the planning tool, NextSat's low-maintenance strategy, and how easy it was for the SRPs to add the activities in ASPEN allowed SRPs to plan for multiple satellites and account for many real-world factors in planning operations.

The operational success of ASPEN's OE model can be largely attributed to the general benefits of automated planning and scheduling in which reusable activity models allow for faster human planning and decrease the need for redundant verification steps in the operations process; high levels of model parameter control allow quick adjustments to be made to both activities and the initial and/or ongoing state of the spacecraft and its domain; further, automated scheduling helps the plan operator or user view the "conflicts" that may or may not exist in a plan. The basic planning constructs of the ASPEN Modeling Language along with more complex capabilities introduced for OE (schema-level uncertainty and recursive decompositions) as well as the method in which the ASPEN core can invoke specialized functions for any existing model, particularly contributed to the success of this application deployment.

From an operational standpoint, long-term planning time could also have been reduced by requesting all visible contacts, instead of creating expected scenarios with their associated contacts; however, the activities of the scenarios would not have been validated to the extent they were long before execution.

Related Work

In June 1997, a docking of a Progress supply ship at the Mir space station was attempted but did not succeed. The Air Force Research Laboratory (AFRL) launched XSS-10 in 2003 and XSS-11 in 2005 with the objectives of advancing autonomous navigation and maneuvering technologies. Orbital Express was the first successful demonstrator of autonomous ORU (Orbital Replacement Unit) transfers in the world and of autonomous refueling in the United States. While several other missions over the past decade have approached the idea of autonomous satellite servicing with rendezvous and other robotic maneuvers, including NASA's Demonstration of Autonomous Rendezvous Technology (DART) satellite and Japan's National Space Development Agency (NASDA) Engineering Test Satellite 7, OE was the first successful demonstrator of autonomous rendezvous and docking (Dornheim 2006).

Planning operations for the Mars Exploration Rover (MER) mission is aided by the NASA Ames Research Center software tool Mixed Initiative Activity Plan Generator (MAPGEN) (AI-Chang et al. 2003), which is similar to ASPEN as an automated planner through the use of activities and temporal constraints. The nature of search for MAPGEN does not allow it to search the infeasible space for plan solutions; that is, when a constraint violation arises, the planner backtracks. ASPEN admits search in the infeasible space (in fact, threats and constraint violations are rolled up into a single generic entity called a conflict) allowing for faster response to off-nominal execution (Chien et al. 2000; Chien et al 2005). In fact, ASPEN has been demonstrated for in situ rover technology (Castano et al. 2007; Estlin et al. 1999; Gaines et al. 2008)

P. Maldague, A. Ko, D. Page, and T. Starbird (Maldague et al. 1997) have developed a schedule generation framework (APGEN) that automatically generates and validates plans used for commanding spacecraft but does not perform general planning and scheduling.

For an even more expansive survey of time line–based planning systems, see the paper by Chien et al. (2012). Note that much of the effort for Orbital Express was not in planning technology per se (we assumed it existed and worked), but in agile planning model generation.

Acknowledgements

Portions of this work were performed by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration. Portions of this work were performed by the Jet Propulsion Laboratory,

California Institute of Technology, and were sponsored by the Defense Advanced Research Projects Agency. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not constitute or imply its endorsement by the United States government, or the Defense Advanced Research Projects Agency, or the Jet Propulsion Laboratory, California Institute of Technology.

References

AI-Chang, M.; Bresina, J.; Charest, L.; Jonsson, A.; Hsu, J.; Kanefsky, B.; Maldague, P.; Morris, P.; Rajan, K.; and Yglesias, J. 2003. MAPGEN: Mixed Initiative Planning and Scheduling for the Mars 03 Mer Mission. Paper presented at the 7th International Symposium on Artificial Intelligence, Robotics, and Automation in Space (i-SAIRAS-03), Nara, Japan, 19–23 May.

Castano, R.; Estlin, T.; Anderson, R.; Gaines, D.; Castano, A.; Bornstein, B.; Chouinard, C.; Judd, M. 2007. OASIS: Onboard Autonomous Science Investigation System for Opportunistic Rover Science. *Journal of Field Robotics* 24(5): 379–397.

Chien, S.; Johnston, M.; Frank, J.; Giuliano, M.; Kavelaars, A.; Lenzen, C.; Policella, N. 2012. A Generalized Timeline Representation, Services, and Interface for Automating Space Mission Operations. Paper presented at the 12th International Conference on Space Operations (SpaceOps 2012), Stockholm, Sweden, 11–15 June.

Chien, S.; Rabideau, G.; Knight, R.; Sherwood, R.; Engelhardt, B.; Mutz, D.; Estlin, T.; Smith, B.; Fisher, F.; Barrett, T.; Stebbins, G.; Tran, D. 2000. ASPEN – Automating Space Mission Operations Using Automated Planning and Scheduling. Paper presented at the 1st International Conference on Space Operations (SpaceOps 2000), Toulouse, France, 19-23 June.

Chien, S.; Sherwood, R.; Tran, D.; Cichy, B.; Rabideau, G.; Castano, R.; Davies, A.; Mandel, D.; Frye, S.; Trout, B.; Shulman, S.; Boyer, D. 2005. Using Autonomy Flight Software to Improve Science Return on Earth Observing One. *Journal of Aerospace Computing, Information, and Communication* 2(4): 196–216.

Dornheim, M. A. Orbital Express to Test Full Autonomy for On-Orbit Service, 2006. *Aviation Week and Space Technology* 164(23) (June 4): 46–50.

Estlin, T.; Mann, T.; Gray, A.; Rabideau, G.; Castano, R.; Chien, S.; and Mjolsness, E. 1999. An Integrated System for Multi-Rover Scientific Exploration. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*. Menlo Park, CA: AAAI Press.

Gaines, D. M.; Estlin, T.; and Chouinard, C. 2008. Spatial Coverage Planning and Optimization for Planetary Exploration. Paper presented at the 9th International Symposium on Artificial Intelligence, Robotics, and Automation in Space (i-SAIRAS 2008), Los Angeles, 26–29 February.

Kortenkamp, D.; Bonasso, R. P.; and Schreckenghost, D. 2008. A Procedure Representation Language for Human Spaceflight Operations. 2008. Paper presented at the 9th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS-08), Los Angeles, 26–29 February.

Maldague, P.; Ko, A.; Page, D.; and Starbird, T. 1997. APGEN: A Multi-Mission Semi-Automated Planning Tool. Paper presented at the 1st International Workshop on Planning and Scheduling for Space. Oxnard, CA, October.

Rabideau, R.; Knight, R.; Chien, S.; Fukunaga, A.; Govindjee, A. 1999. Iterative Repair Planning for Spacecraft Operations in the ASPEN System. Paper presented at the International Symposium on Artificial Intelligence, Robotics, and Automation in Space, Noordwijk, The Netherlands, 1–3 June.

Russell Knight is a member of the technical staff, Artificial Intelligence Group, Jet Propulsion Laboratory, California Institute of Technology. He is currently working on the automated scheduing and planning environment (ASPEN) planning system for mission operations, the Continuous Activity Scheduling, Planning, Execution, and Replanning (CASPER) real-time planning system, and the Mission Data System (MDS) (specifically the scheduler). Additionally, he supports research efforts in Antarctica. His research interests are in planning and scheduling, heuristic search, operations research, and space exploration automation.

Caroline Chouinard joined Red Canyon Software in 2011 during the prelaunch software verification stage of the Juno mission. She is currently working on the next-generation astronaut transport capsule known as MPCV (multipurpose crew vehicle). Chouinard came to Red Canyon Software from the Jet Propulsion Laboratory where she was most recently leading the Sequence Virtual Team for Cassini. She also held multiple roles on the Mars Exploration Rovers mission in software development and both strategic and tactical mission operations and was awarded for her work on the DARPA-funded technology demonstration Orbital Express.

Grailing Jones, Jr., is a senior technical staff member in the Planning and Execution Systems Group at the Jet Propulsion Laboratory. He has developed and operated mission planning and execution systems for a variety of NASA and DOD projects. He earned his B.S. in astronautics from the United States Air Force Academy, master's in aerospace engineering from the University of Colorado at Boulder, and master's in systems architecture and engineering from the University of Southern California.

Daniel Tran is a member of the technical staff in the Artificial Intelligence Group at the Jet Propulsion Laboratory, California Institute of Technology, where he is working on developing automated planning and scheduling systems for on-board spacecraft commanding. Tran attended the University of Washington and received a B.S. in computer engineering. He is currently the software lead for the Autonomous Sciencecraft Experiment, and he is cowinner of the 2005 NASA Software of the Year award.

Enhanced Telemetry Monitoring with Novelty Detection

José Martínez Heras, Alessandro Donati

■ *Typically, automatic telemetry monitoring in space operations is performed by out-of-limits (OOL) alarms. This approach consists of defining an upper and lower threshold so that when a measurement goes above the upper limit or below the lower one, an alarm is triggered. We discuss the limitations of the out-of-limits approach and propose a new monitoring paradigm based on novelty detection. The proposed monitoring approach can detect novel behaviors, which are often signatures of anomalies, very early — allowing engineers in some cases to react before the anomaly develops. A prototype implementing this monitoring approach has been implemented and applied to several ESA missions. The operational assessment from the XMM-Newton operations team is presented.*

The most widely extended approach for automatically detecting anomalous behavior in space operations is the use of out-of-limits (OOL) alarms. The OOL approach consists of defining an upper and lower threshold so that when a measurement goes above the upper limit or below the lower one, an alarm is triggered. Then engineers will inspect the parameter that is out of limits and determine whether it is an anomaly or not and decide which action to take (for example, run a procedure). This is the original out-of-limits concept.

The current OOL concept has evolved to cope with more situations such as distinguishing between soft and hard limits; for example, a soft OOL triggers a warning to pay attention, a hard OOL triggers an error that demands attention. Soft limits are contained within hard limits. In addition OOL thresholds (soft and hard) can be configured so that different thresholds are applicable in different situations (for example, depending on the working mode of a given instrument).

 ISSN 0738-4602

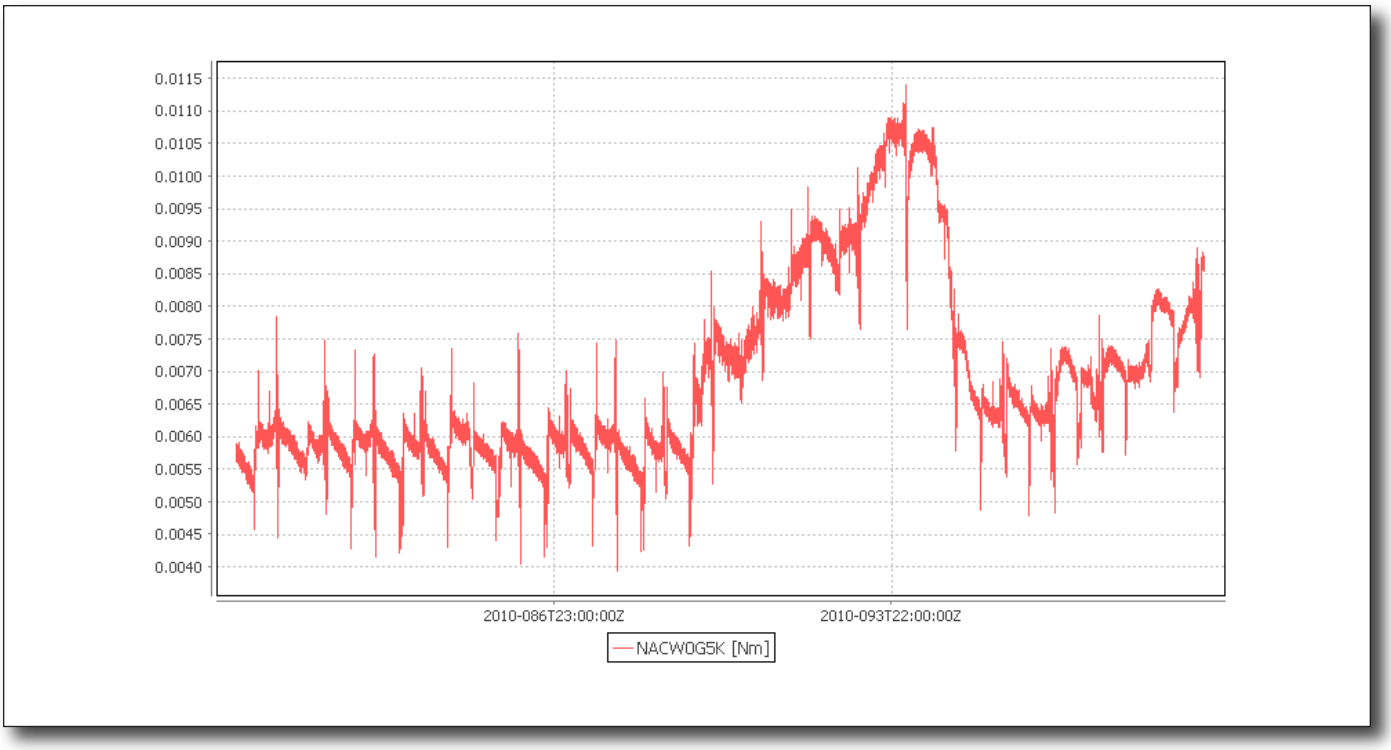


Figure 1. A Typical Example of Anomalous Behaviors.

Venus Express reaction wheel 4 friction increases starting in day of year 2010.89. This behavior was not detected by out-of-limits alerts as the upper limit was set to 0.02 at that time. In this case, this behavior was recognized by Venus Express engineers as they closely monitored the reaction wheels even if they did not trigger any out-of-limits alerts.

While out-of-limits approaches are useful and they successfully trigger alarms when parameter readings go out the defined thresholds, they suffer from some limitations. First, some behaviors are anomalous even if they are within the defined limits. A typical example is shown in figure 1. The parameter values reach the upper limit but do not hit it. Since engineers don't know when this will happen they have to monitor key telemetry parameters closely even if in most cases everything would be nominal. Paradoxically, sometimes the anomalous behavior is more in limits than the nominal one. Figure 5, depicted later on in this article, shows an example of this situation. More information about this anomaly will be discussed later. Second, OOL bounds are not defined for every parameter. Engineers only define OOL for a subset of parameters for which they want to receive alarms if they exceed the limits. Therefore, OOL is not systematic in the sense that it does not cover every parameter. Third, and quite often, engineers receive OOL alarms that are completely expected. A typical example is the OOL bounds defined for the automatic control gain (AGC) during a pass. At acquisition of signal (AOS) and loss of signal (LOS) the AGC goes outside limits. However, it is expected to happen and in every pass these two OOL alarms will be raised. Ideally, engineers should only have to investigate real potential anomalies. Finally, it requires effort to adapt OOL alerts to useful values as the mission goes through different phases or simply degrades with time.

The novelty detector project has been developed to cope with the current OOL limitations. The novelty detector main goal is to automatically detect anomalies and report them to engineers for further investigation. The ideal novelty detector should take into account that parameters can behave nominally in several different ways. In addition, it should not make any assumption on what kind of behavior or how many different behaviors a parameter will have. This will allow it to work with any parameter without the need of prior knowledge.

Approaches to Automatic Anomaly Identification

There are mainly three approaches to identify novel behavior (possibly anomalies): supervised, unsupervised, and semisupervised. The supervised approach consists of having labeled examples of both nominal and anomalous behavior. This approach works quite well if we need to recognize previously labeled nom-

inal and anomalous behavior. Its major drawback is that it can only identify anomaly occurrences for the anomaly types that it knows already. This is a major limitation since the anomalies that generally will affect operations the most are the ones that are happening for the first time. Being able to recognize first-time-ever anomalies quickly allows flight control teams to take action immediately.

The unsupervised approach consists of having unlabeled examples of data — no prior knowledge is provided. The implicit assumption made by the systems using a nonsupervised approach is that anomalies happen far less often than nominal behaviors. So they attempt to automatically distinguish what is nominal and what it anomalous. The major drawback is the risk of missing anomalies: if an anomaly happened several times in the past, a nonsupervised system may consider it a normal behavior and not report it in the future.

The semisupervised approach is a combination of the supervised and unsupervised approaches. It consists of providing only nominal behaviors examples. The advantages of this approach are that engineers are in full control to specify what should be considered as nominal, repeated anomalies can be detected since they are not in the nominal set, and since no assumptions are made about the possible behavior of the anomalies, any anomalous behavior can be potentially detected.

The proposed monitoring paradigm follows a semisupervised approach to perform anomaly detection. We will use the term *novelty detection* instead of *anomaly detection* since the only thing that can be said is that a behavior is novel when compared to a set of behaviors known to be nominal. The new behavior might well be also nominal but so far not present in the nominal set. The decision of classifying a new behavior as nominal or anomalous is left to the flight control engineers.

Novel Behavior Detection

To characterize behaviors we compute four statistical features (average, standard deviation, maximum, and minimum) of fixed-length periods. The duration of the time period is chosen so that it represents a natural time span (for example, orbit period or time covered by the short-term planning). The exact duration is not critical; however, it should be long enough to allow behaviors to develop and not so long that many different behaviors happen in it.

While there are other ways to characterize behavior in a given time period (for example, Fourier transformations, wavelets, and so on) we used statistical features because they are robust to sampling rate changes and behavior order, and work even if very few samples are available. In addition, they are compatible with the future European Space Operations Centre (ESOC) infrastructure data archive (DARC). DARC precomputes and make available these statistical features.

Figure 2 shows representation of how these fixed time periods look in a side-by-side two dimensional comparison. We are showing this representation in this document only as an example. In reality, four dimensions (average, standard deviation, maximum, and minimum) are used simultaneously.

Periods Distance

Once we have defined the representation of a time period for a given parameter we need to be able to compare time periods. We need a distance measurement so that we can say that for a given parameter *A,* the period *X* is closer to the period *Y* than to the period *Z*. Mathematically: $d(X, Y) < d(X, Z)$. We use the Euclidean distance as distance measurement (equation 1):

$$d(X,Y) = \sqrt{(avg_x - avg_y)^2 + (std_x - std_y)^2 + (\max_x - \max_y)^2 + (\min_x - \min_y)^2}$$

Outlier Detection

We make use of outlier detection techniques to find which periods have anomalous behavior. The general assumption is that anomalous behaviors will have greater distances to known nominal behaviors than known nominal behaviors among them. The question is how big the distance should be so that it can be considered a novel behavior. If the distance is too small many false anomalies will be reported. If the distance is too big then some anomalies will be missed.

The solution to overcome the problem of having to define an outlier distance is to use local density outlier detection techniques. The most widely used is called local outlier factor (LOF) (Breunig et al. 2000). LOF computes a factor that gives an indication of the degree of outlierness (novel behavior). It takes into account the density of the *k* closest points. If they are very dense, little distance is required to consider a new behavior a novelty. If the neighbors are sparse a bigger distance is required to consider a new behavior a novelty.

The major disadvantage of LOF is that the resulting factor values are quotient values and hard to interpret. A value of 1 or less indicates a clear inlier, but there is no clear rule for when a point is an outlier. In one data set, a value of 1.1 may already be an outlier; in another data set and parameterization (with strong local fluctuations) a value of 2 could still be an inlier. These differences can also occur within a data set due to the locality of the method.

To overcome the LOF limitations we will use local outlier probabilities (LoOP) (Kriegel et al. 2009). It is a relatively new method derived from LOF. It uses inexpensive local statistics to become less sensitive to the choice of the parameter *k*. In addition, the resulting values are scaled to a value range of [0:1] (Kriegel et al. 2009), that can be directly interpreted as the

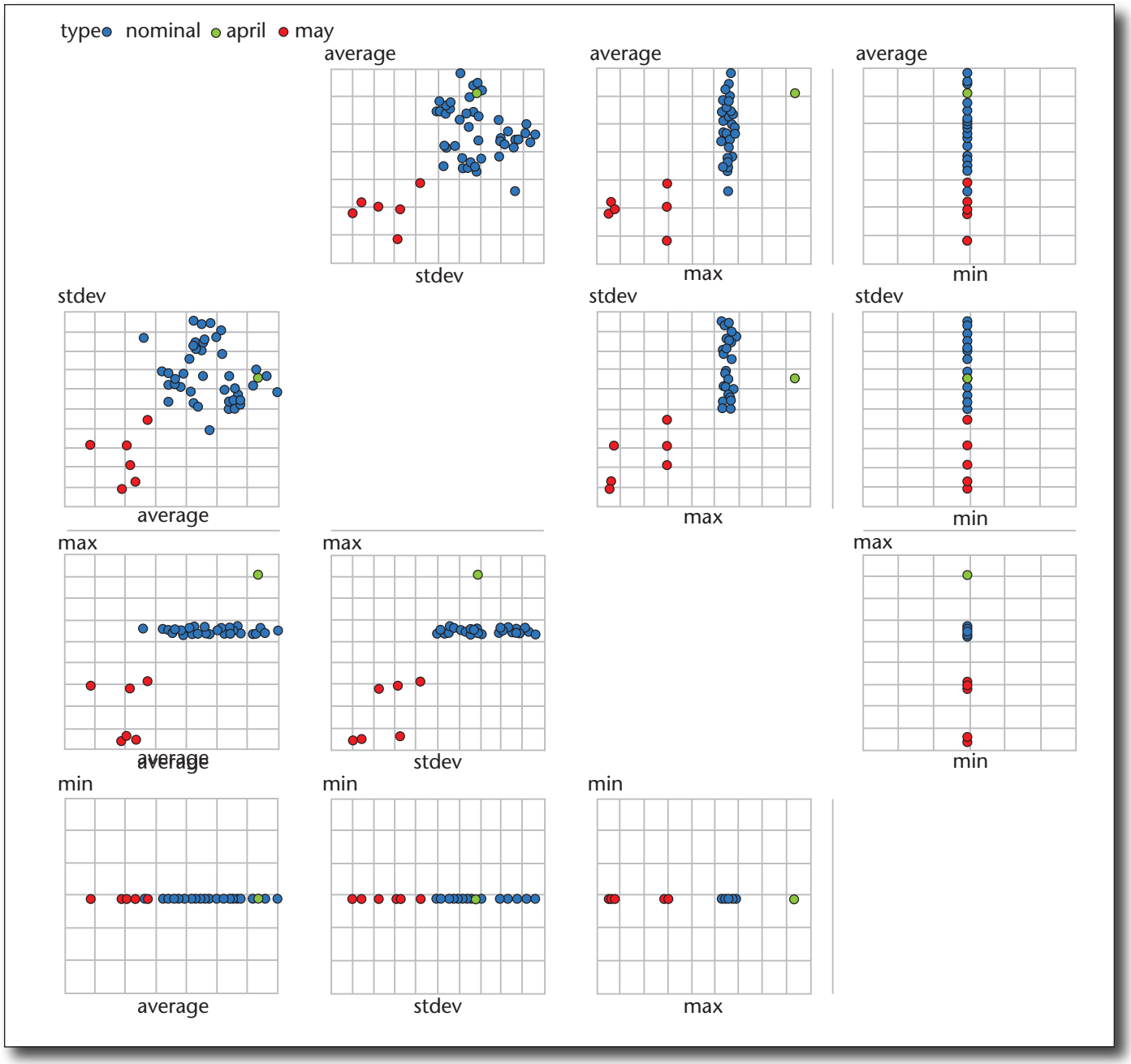


Figure 2. Statistical Features.

Feature extraction and side-by-side comparison of the XMM anomaly shown in figure 5. Every point represents a time period; its position in the chart is given by its statistical features. Legend: blue consists of nominal periods (1 January 2009 – 31 March 2009), green represents a spike in April, and red represents the thermostat dithering anomaly3. This two-dimensional example is only an easy way to visualize this representation. In reality, four dimensions (average, standard deviation, maximum, and minimum) are used simultaneously.

probability of being a new behavior. Figure 3 shows an example using two dimensions (for example, average and standard deviation). LoOP has the advantage of being more robust and providing a much more intuitive output. By using LoOP we can rank novel behaviors by novelty probability showing first the behaviors with higher chances to be truly novel.

Novel Behavior Detection — Summary

We use LoOP (Kriegel et al. 2009) to find whether a new behavior is novel with respect to a set of given behaviors. If the set of given behaviors consists of nominal behaviors only, and LoOP finds that the

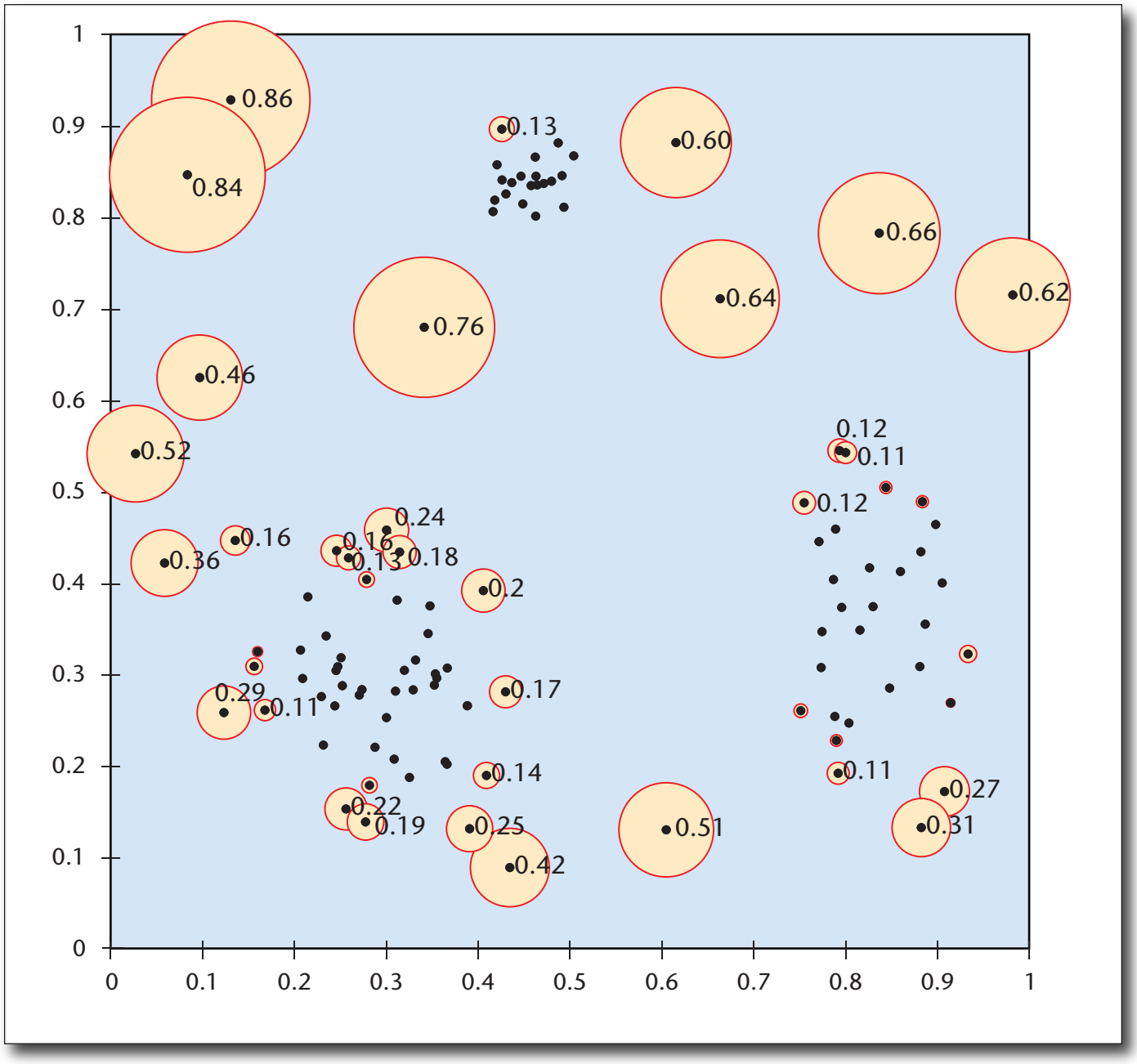


Figure 3. LoOP.

Local outlier probabilities outlier detection technique graphical example with two dimensions. Points with high outlier probability indicate a potential novel behavior.

new behavior is an outlier with high probability, it can mean only two things: the new behavior is either an anomaly or it is nominal in a new way (for example, it behaves differently than other nominal behaviors in the nominal set). The more nominal time periods the nominal set has, the more the chances that the reported novelties are really anomalies.

Behaviors are characterized by simple statistical features (average, standard deviation, maximum, and minimum) in a fixed-size time period (for example, duration of an orbit). This means performing LoOP in four dimensions. Euclidean distance is used to measure how different two time periods are. The same procedure is applied to all parameters, and those with high probability of being novelties are notified to engineers.

To be mode independent of the choice of the parameter k (number of closest points to compute density) we try several $k = \{10, 20, 30\}$ and use as outlier probability the minimum value of these differ-

ent runs. The assumption is that if a new behavior is really an outlier it should be an outlier independently of the number of closest points (k) used to compute the local density. This procedure minimizes the chances of getting false alarms. It has limitations, however, because this approach does not consider novelties in the combination of two or more parameters; it works on a parameter-by-parameter basis only. It is, however, systematic in the sense that it can be applied to every parameter.

Prototype

If flight control engineers would be able to look every day at every parameter they would be able to identify all novelties. Unfortunately, they cannot. There are way too many parameters (in the order of several thousands) and the trend is that this number will increase in future missions. The objective of this prototype is to automate the process of noticing novel behavior at the parameter behavior level.

New behaviors are often signatures of anomalies either happening now or in the way to develop. Noticing them early is of utmost importance for planning corrective measurements and keeping the spacecraft healthy. We should take into account that not every new behavior corresponds to an anomaly: it could be related to a new environmental condition (for example, extremely high radiation) or be totally expected as the result of planned new operations (for example, Venus orbit insertion).

The functionality of being able to automatically detect anomalies has been the driver for this project. However, we understood that we could not build such system. The closest we can get is identifying a new behavior as novel when compared to a set of known behaviors. Hence the name novelty detection.

In order to get closer to our goal of being able to automatically detect anomalies, we choose the known behavior set so that it only contains nominal behaviors. With this configuration, when a new behavior is classified as novel it can only mean two things: it is either an anomaly or a new nominal behavior. As time passes, the set of known nominal behaviors will grow. This has a positive impact in reducing the number of novelty alerts, as many behaviors will be classified as nominal.

The novelty detection prototype makes use of mission utilities and support tools (MUST) (Martínez-Heras, Baumgartner, and Donati 2005; Baumgartner et al. 2005) as housekeeping telemetry and ancillary data provider. The MUST's performance allows the performance of advanced monitoring with novelty detection efficiently.

Functionalities

We will now discuss the two major functionalities of the novelty detection prototype. The underlying principle is the same, but one can achieve one functionality or the other depending on which configuration is used.

Identification of Potential Anomalies

The main purpose of the novelty detection prototype is to detect potential anomalies. For fulfilling this objective we will use as a known periods set the collection of all known nominal behaviors. This way, when a new behavior is classified as novel with a high probability, it is very likely that it would be an anomaly. It could be still a new kind of nominal behavior but, as time passes, this should happen less and less frequently.

Verification of Expected New Behavior

In addition to identification of potential anomalies, the same novelty detection technique can be used to verify the occurrence of an expected new behavior. For instance, let's say that certain behavior is expected as a consequence of a maneuver and we would like to verify it. A way of doing it with the novelty detection prototype is by using the recent past as the known behavior set. The output of the novelty detection will be the novel behaviors compared with the recent past. These novelties should contain the expected new behaviors and possibly other parameters. These parameters that were not initially foreseen can be considered side effects of the expected behaviors.

Input

Two inputs are required to run the novelty detection prototype: periodicity and the set of known behaviors.

Periodicity is the statistical features needed to characterize a fixed-length time period and has to be computed over a large enough time period. A typical example is to use the periodicity of the orbit or the amount of time that the short-term planning covers.

Set of known behaviors: the novelty detection will detect whether a new behavior is novel as compared with the set on known behaviors specified as input. Two options are recommended: (1) use all nominal behaviors: this is ideal to perform anomaly detection; and (2) use the recent past (this should be used to verify expected new behavior).

Output

The output consists of a text file that contains the list of parameters that are believed to be novel. They are grouped by period and by novelty probability within periods. Figure 4 shows an example of such an output file. The same information is stored in a database for further analysis by client applications.

Operational Validation and Current Usage

To prove the feasibility of the monitoring paradigm with novelty detection we applied it to an already documented anomaly that the ESA satellite XMM-

```
0.935057, no-Gaps, 2010-10-14T00:00:00Z [2010.287.00.00.00], NTTX1000, Loop Error Term. 1
0.817176, no-Gaps, 2010-10-14T00:00:00Z [2010.287.00.00.00], NACW0G15, RW4_SWR estimat friction
0.935141, no-Gaps, 2010-10-15T00:00:00Z [2010.288.00.00.00], NTTX1000, Loop Error Term. 1
0.804666, no-Gaps, 2010-10-15T00:00:00Z [2010.288.00.00.00], NACW0R08, AUT_GUID Cmd Quater Vy
```

Figure 4. Example Output File.

Novelties found for Venus Express using the novelty detection prototype for verifying expected new behavior. The format of the file output is the probability of being an outlier, whether this parameter had data gaps during this period, the start time of the period (in two time formats), parameter mnemonic, and parameter description.

Newton experienced in 2009 and checked when the novelty detection prototype would have been able to detect the anomaly. Here is an excerpt of the paper that describes the anomaly and measures taken by the FCT (flight control team) to cope with it (Pantaleoni et al. 2010):

> We noticed that the thermostat T6073 started to have a strange behavior since mid-May 2009, already 2 months before the failure was spotted. The thermostat range reduced the temperature where it opened and started to decrease, a sign of a deterioration of the high threshold, even if the bottom limit was respected quite well, until mid-July, when the upper limit and the lower limit went very close to each other. The thermostat started to oscillate, in a narrow temperature range, until it did not close anymore at the correct temperature, and it let the temperature go down to almost 22 deg. This first temperature drop was not spotted because it did not generate any OOL. After that the thermostat had some cycles with a nominal behavior, but on 13 July 2009 the temperature went down deeper, to 21.25, triggering an OOL and allowing the FCT to spot the problem.

We configured the novelty detection prototype to consider data in the range (January 2009, March 2009) as nominal. We used as time period 48 hours since it is the duration of an XMM-Newton orbit. Then we run the novelty detection prototype for the period (April 2009, July 2009). The results is that the novelty detection prototype managed to find unusual behavior 2 months before the out-of-limit triggered. This is remarkable not only because it allows to react to anomalies early, but also because it matches flight control engineers diagnosis results and mimics the effect of having somebody looking every day at every parameter and noticing if something new is happening. Figure 5 shows where the OOL triggered and where the novel behavior was found.

Another tests related with caging was performed with the novelty detection prototype. In the XMM FCT's words:

> XMM reaction wheel 1 faces unexpected increases in torque and current consumption during stable pointing. The reaction wheel 1 of XMM suffers from an increment of friction. The possible cause is cage instability due to under-lubrication, *Rosetta,* another spacecraft, also suffered from the same problem. The Rosetta manufacturer pointed to under-lubrication as a possible cause. Since this anomalous phenomenon has been spotted, the damaged reaction wheel has been closely monitored. The XMM FCT wanted to know if the novelty detection could have been able to detect the caging earlier than the flight control team did?
>
> For this test, nominal data was defined as the month of May 2005 when no caging was present. May 2010 was investigated for caging on parameter A5249. The parameter A5249 refers to the torque of the reaction wheel 1. Caging has effectively been detected and the probabilities computed are high enough to be seriously considered. In the end, if the XMM flight control team could have used novelty detection beforehand, the caging phenomenon would have been detected and closely monitored earlier. The lifetime of the damaged wheel could have been saved thanks to an earlier relubrication.

Currently, the novelty detection prototype checks every day around 2000 XMM-Newton housekeeping telemetry parameters and reports which of them, if any, has a new behavior. The results are sorted by probability of certainty of being a new behavior. The novelty detection for XMM is integrated in a wider scope project, XMM early warning system (XEWS) (Kirsch et al. 2012). XEWS is developed to perform near real-time trend analysis of spacecraft parameters in order to detect early degradation of components. XEWS will enable the mission to perform early countermeasures in case degradation is detected.

In addition, the novelty detection prototype has been integrated as a plug-in of WebMUST (Oliveira et al. 2012). WebMUST is a web-based interface to access telemetry and ancillary data in the MUST (Martínez-Heras et al. 2005, Baumgartner et al. 2005) Repository. WebMUST allows users to very efficiently create plots and reports. WebMUST can also sup-

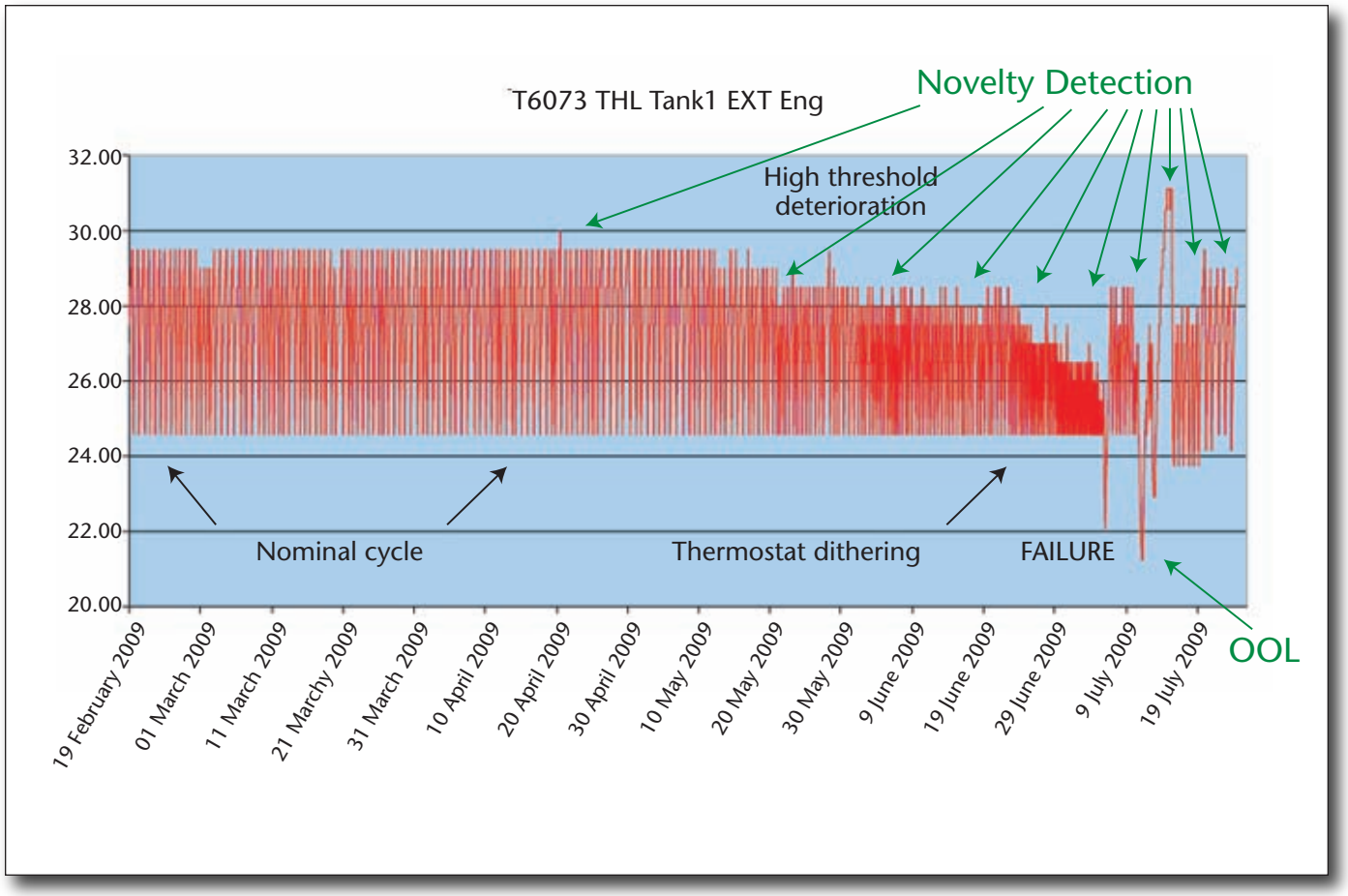


Figure 5. Nominal Behavior.

February – April, OOL on 13 July. This thermostat has been properly working showing the same behavior for 10 years. However, it started to have a strange behavior since mid-May 2009 and it was only noticed two months after (July 2009) when it crossed the lower limit. For this type of anomaly, the out-of-limits checks are not effective because, paradoxically, the behavior of the anomaly was "more in limits" than before. The proposed novelty detection monitoring technique could find this anomaly two months before the out-of-limit alarm triggered.

port characterizations and anomaly investigation using the DrMUST (Martínez-Heras et al. 2012, Martínez-Heras et al. 2009) plug-in. A screen capture of the novelty detection display for an expected new behavior is shown in figure 6.

Related Work

The problem of automatically finding unusual behavior has been addressed by other researchers in a number of fields, both space and nonspace.

For instance, A. Patterson-Hine and colleagues (Patterson-Hine et al. 2001) uses a model-based approach to detect anomalies in helicopters; however, model-based approaches require a big upfront engineering effort. In this project we have focused on approaches that require as little engineering effort as possible.

E. Keogh and colleagues (Keogh, Lin, and Fu 2005) describe the algorithm HOT SAX in order to automatically and efficiently find time series discords. Time series discords are defined as subsequences of longer time series that are maximally different to all the rest of the time series subsequences. Its major advantage is its simplicity as it just requires a single input: the length of the subsequence (in our case it would be the period length). While HOT SAX is successful at finding the ranking of discords for time series, it is of little use to spacecraft engineers that need to understand if these discords are relevant or not, especially when they are monitoring thousands of parameters of different nature. In addition, it is difficult to know how important a top-ranked discord of a certain parameter is in relation to the other parameters' top-ranked discords.

D. L. Iverson and colleagues (Iverson et al. 2012) present the inductive monitoring system (IMS), which uses clustering to characterize normal itera-

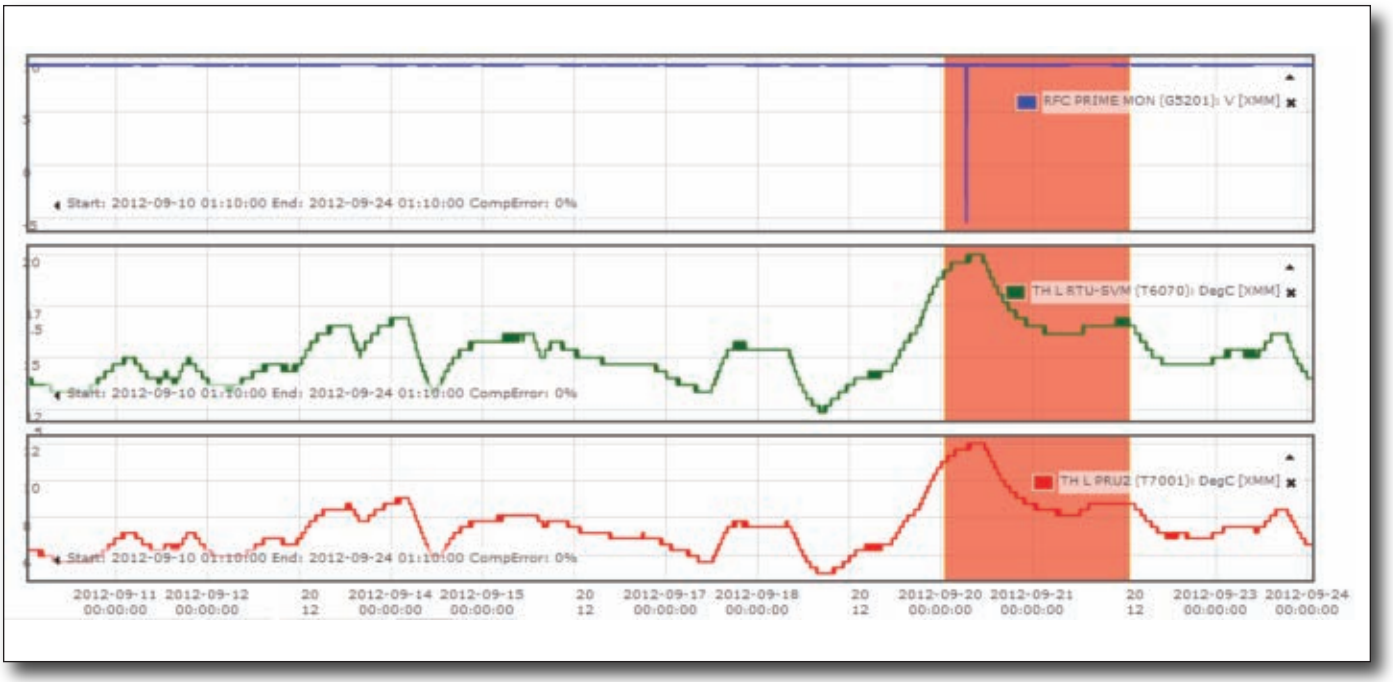


Figure 6. Novelty Detection Display for an Expected New Behavior.

Screen capture of the novelty detection plug-in integrated into WebMUST. It shows an expected new behavior for XMM. The highlighted area corresponds to the period when the novel behavior was detected.

tions between parameters. Engineers provide a vector describing which measurements make sense to monitor together (for example, pressures, valve positions, temperatures) and examples of nominal behavior. Then IMS builds clusters based in the nominal data. The learning process is influenced by three learning parameters: the maximum cluster radius, the initial cluster size, and the cluster growth percent. After the clusters are created, the IMS performs monitoring by comparing the current vector data with the clusters database. If the distance is 0 or close to 0, its behavior is considered nominal. As the distance increases, the current behavior can be considered more anomalous. The clear advantage of IMS over the proposed novelty detection monitoring technique is that it can detect unusual behaviors in a combination of parameters. However, it has some disadvantages with respect to the proposed novelty detection technique: the grouping of which parameters need to be monitored together, apart from requiring engineering effort, determines the kind of anomalies the system will be able to detect. In this work we are concerned with the detection of novel behaviors as soon as possible even if engineers do not think this will be possible. Another disadvantage is that the anomaly score is not intuitive; if it is 0 or close to 0 it is nominal, but it is not clear when engineers should start paying attention and performing investigations. A further disadvantage is the amount of tuning that is required to have the IMS work properly: the weight in the vector components and the values of the three learning parameters have a strong impact in the creation of the cluster database and, therefore, in the results from the monitoring phase.

Conclusions

We have introduced a new monitoring paradigm based on novelty detection. In this approach, every day every telemetry parameter is automatically scanned and a list of the parameters that exhibit a novel behavior is reported to flight control engineers. New behaviors are often signatures of anomalies either happening now or in the way to develop. Noticing them early is of utmost importance for planning corrective measurements and keeping the spacecraft healthy.

A clear advantage of the proposed monitoring paradigm is the little amount of engineering effort required: the only inputs required consist of the period duration (for example, 1 day) and ranges of times to be used as examples of nominal behavior. During the validation phase, users really appreciated that it generates very few false alarms: the fact that it uses a local density outlier detection technique avoids the need of using a distance threshold to detect new behaviors. Therefore, this approach does not suffer from the problem of having to define a threshold

where having it too small would lead to many false alarms and having the threshold too big will lead to missing new behaviors.

The proposed novelty detection monitoring approach has been successfully validated with XMM anomalies, finding them before they were triggered by the out-of-limits alarms, sometimes as early as two months in advance. Currently, the XMM mission uses the novelty detection prototype to detect new behaviors on about 2000 parameters on a daily basis. We would like to highlight that monitoring with novelty detection is not mission specific but generic. We can easily adapt it to any ESOC controlled mission since we use MUST (Martínez-Heras et al. 2005, Baumgartner et al. 2005) as the data provider.

We believe that every mission will benefit from the adoption of the novelty detection monitoring paradigm as complement to the classic out-of-limits mechanism. Being able to know which few parameters (out of several thousands) exhibit a new behavior helps flight control engineers to efficiently direct their monitoring efforts. The ESA's patents group has decided to protect the proposed monitoring paradigm by filing an international patent (WO2013 010569).

References

Baumgartner, A.; Martínez-Heras, J.; Donati, A.; Quintana, M. 2005. MUST — A Platform for Introducing Innovative Technologies in Operations. Paper presented at the 2005 International Symposium on Artificial Intelligence, Robotics and Automation for Space, Munich, Germany, 5–9 September.

Breunig, M. M.; Kriegel, H.-P.; Ng, R. T.; Sander, J. 2000. LOF: Identifying Density-Based Local Outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data* (ACM SIGMOD Record), 93–104. New York: Association for Computing Machinery. dx.doi.org/10.1145/342009.335388

Iverson, D. L.; Martin, R.; Schwabacher, M.; Spirkovska, L.; Taylor, W.; Mackey, R.; Castle, J. P.; Baskaran, V. 2012. General Purpose Data-Driven Monitoring for Space Operations. *Journal of Aerospace Computing, Information, and Communication* 9(2): 26–44. In *Proceedings of the 5th IEEE International Conference on Data Mining* (ICDM 2005), 226–233. Piscataway, NJ: Institute of Electrical and Electronics Engineers. dx.doi.org/10.1109/ICDM.2005.79

Keogh, E.; Lin, J.; and Fu, A. 2005. HOT SAX: Efficiently Finding the Most Unusual Time Series Subsequence. In *Proceedings of the 5th IEEE International Conference on Data Mining* (ICDM 2005), 226–233. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Kirsch, M.; Martin, J.; Pantaleoni, M.; Southworth, R.; Schmidt, F.; Webert, D.; Weissmann, U. 2012. Cage Instability of XMM-Newton's Reaction Wheels Discovered During the Development of an Early Degradation Warning System. Paper presented at the 12th International Conference on Space Operations (SpaceOps 2012), Stockholm, Sweden, 11–15 June.

Kriegel, H. P.; Kröger, P.; Schubert, E.; and Zimek, A. 2009. LoOP: Local Outlier Probabilities. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management,* 1649–1652. New York: Association for Computing Machinery. dx.doi.org/10.1145/1645953.1646195

Martínez-Heras, J.; Baumgartner, A.; Donati, A. 2005. MUST: Mission Utility and Support Tools. Paper presented at the Data Systems in Aerospace Conference (DASIA 2005), Edinburgh, Scotland, 30 May–2 June.

Martínez-Heras, J. A.; Donati, A.; Sousa, B.; Fischer, J. 2012. DrMUST — A Data Mining Approach for Anomaly Investigation. Paper presented at the 12th International Conference on Space Operations (SpaceOps 2012), Stockholm, Sweden, 11–15 June.

Martínez-Heras, J. A.; Yeung, K.; Donati, A.; Sousa, B.; Keil, N. 2009; DrMUST: Automating the Anomaly Investigation First-Cut. Paper presented at the IJCAI-09 Workshop on Artificial Intelligence in Space. Pasadena, California, USA, 17–18 July.

Oliveira, H.; Lais, A.; Francisco, T.; Donati, A. 2012. Enabling Visualization of Large Telemetry Datasets. Paper presented at the 12th International Conference on Space Operations (SpaceOps 2012), Stockholm, Sweden, 11–15 June.

Pantaleoni, M.; Kirsch, M.; Martin, J.; Weissmann, U.; Krusenstiern, N. 2010. The New Operational Strategy for XMM-Newton's Tank Thermal Control, After Low Temperature Protection Thermostat Failure. Paper presented at the 10th International Conference on Space Operations (SpaceOps 2010), Huntsville, AL, 25–30 Apr.

Patterson-Hine, A.; Hindson, W.; Sanderfer, D.; Deb, S.; Domagala, C. 2001. A Model-Based Health Monitoring and Diagnostic System for the UH-60 Helicopter. In *Proceedings of the American Helicopter Forum,* Volume 57, No. 2, 1552–1564. Alexandria, VA: American Helicopter Society, Inc.

José Martínez-Heras is a software engineer/researcher at the Advanced Operations Concepts Office at the European Space Agency. His main interest consists of enhancing monitoring and diagnostics in space operations. His research includes the enhancement of remote observability, availability of space data, reduction of information overload, and data mining and machine learning applied to space opearations. In recent years he was the main coauthor of four patents related to data compression, diagnostics, and monitoring. Martínez-Heras holds a master of computer science engineering degree from the Univerity of Málaga, Spain.

Alessandro Donati has 25 years of experience in satellite operations at the European Space Operations Centre of ESA in Darmstadt, Germany. In 2001 he founded and has since led a skunk-work style managed team devoted to introducing and exploiting advanced enabling technology in support of innovative mission operations concepts. The operational tasks covered by the applied research include planning, scheduling, diagnostic, resources management, behavior modeling, and forecasting. The technologies of interest include artificial intelligence, data mining, and soft computing, among others. In recent years he was coauthor of three patents in the area of data compression and diagnosis. Donati holds a master of electronic engineering degree from the University of Rome La Sapienza.

Science Autonomy for Rover Subsurface Exploration of the Atacama Desert

David Wettergreen, Greydon Foil, Michael Furlong, David R. Thompson

■ *As planetary rovers expand their capabilities, traveling longer distances, deploying complex tools, and collecting voluminous scientific data, the requirements for intelligent guidance and control also grow. This, coupled with limited bandwidth and latencies, motivates on-board autonomy that ensures the quality of the science data return. Increasing quality of the data requires better sample selection, data validation, and data reduction. Robotic studies in Mars-like desert terrain have advanced autonomy for long-distance exploration and seeded technologies for planetary rover missions. In these field experiments the remote science team uses a novel control strategy that intersperses preplanned activities with autonomous decision making. The robot performs automatic data collection, interpretation, and response at multiple spatial scales. Specific capabilities include instrument calibration, visual targeting of selected features, an on-board database of collected data, and a long-range path planner that guides the robot using analysis of current surface and prior satellite data. Field experiments in the Atacama Desert of Chile over the past decade demonstrate these capabilities and illustrate current challenges and future directions.*

Robotic explorers communicate only intermittently with scientists because of limited opportunities for visibility by Earth-based antennas and the growing number of spacecraft needing attention. The data rate of deep space communication is also very limited. Autonomy can significantly improve science productivity in intervals between communication opportunities. In particular, science autonomy employs on-board analysis to make decisions affecting the scientific measurements that will be collected or transmitted.

We define *science autonomy* as using information about science objectives and interpretation of science instrument data to determine rover actions. Science autonomy encompasses detection and intelligent selection of measurements and samples, automatic acquisition of measurements. This includes automated approach and instrument/tool placement as well as calibration and verification, meaning collecting the intended measurement or sample. Intelligent collection of scientific measurements can increase both the quantity and quality of information gathered. Science autonomy describes utilizing scientific information to guide rover actions, for example, to execute an intelligent survey

 ISSN 0738-4602

Figure 1. Zoë in the Atacama Desert.

or mapping strategy that adapts as data is collected. Decisions about which samples to acquire and where and when to travel next can be based upon metrics of information gain. Similar metrics can also be used to prioritize science data for download. Intelligent compression strategies use knowledge or models of content to interpret and summarize in a compact form. The ultimate goal of science autonomy is to embody sufficient understanding, quantified by models and metrics, so that rovers can independently choose actions that best support the scientific investigation in which they are engaged. Rovers will take their goals and guidance from scientists, but when isolated they should make scientifically rational decisions and when in communication they should provide the most relevant information possible.

Science autonomy is especially valuable for surface rover operations because missions have finite lifetime and rarely revisit sites after the first encounter — the rover must make good decisions and get it right the first time. Recent demonstrations on spacecraft show increasingly sophisticated science autonomy capabilities. Milestones include target tracking during the Deep Impact comet flyby (Mastrodemos, Kubitschek, and Synnott 2005); target detection and response by the Mars Exploration Rovers (Castaño et al. 2008; Estlin et al. 2012); and spectral detection, discovery, and mapping by the EO-1 spacecraft (Chien et al. 2005; Davies et al. 2006; Doggett et al. 2006; Ip et al. 2006; Thompson et al. 2013). At the same time, new smart instruments are beginning to incorporate autonomous science data analysis directly (Wagstaff et al. 2013) and provide information that can be used to guide the rovers' targeting and operation.

These techniques and others will enable surface rovers to achieve multiday autonomous operations. Currently multiday rover plans do not travel over the horizon of yesterday's imagery, which limits the daily science yield. However, rover navigation already permits safe over-the-horizon traverses, and in principle a rover could autonomously survey large areas of terrain with its full suite of instruments. In one natural arrangement, operators would direct the rover using waypoints determined from satellite images, relying on rover autonomy for low-level hazard avoidance and science target selection en route.

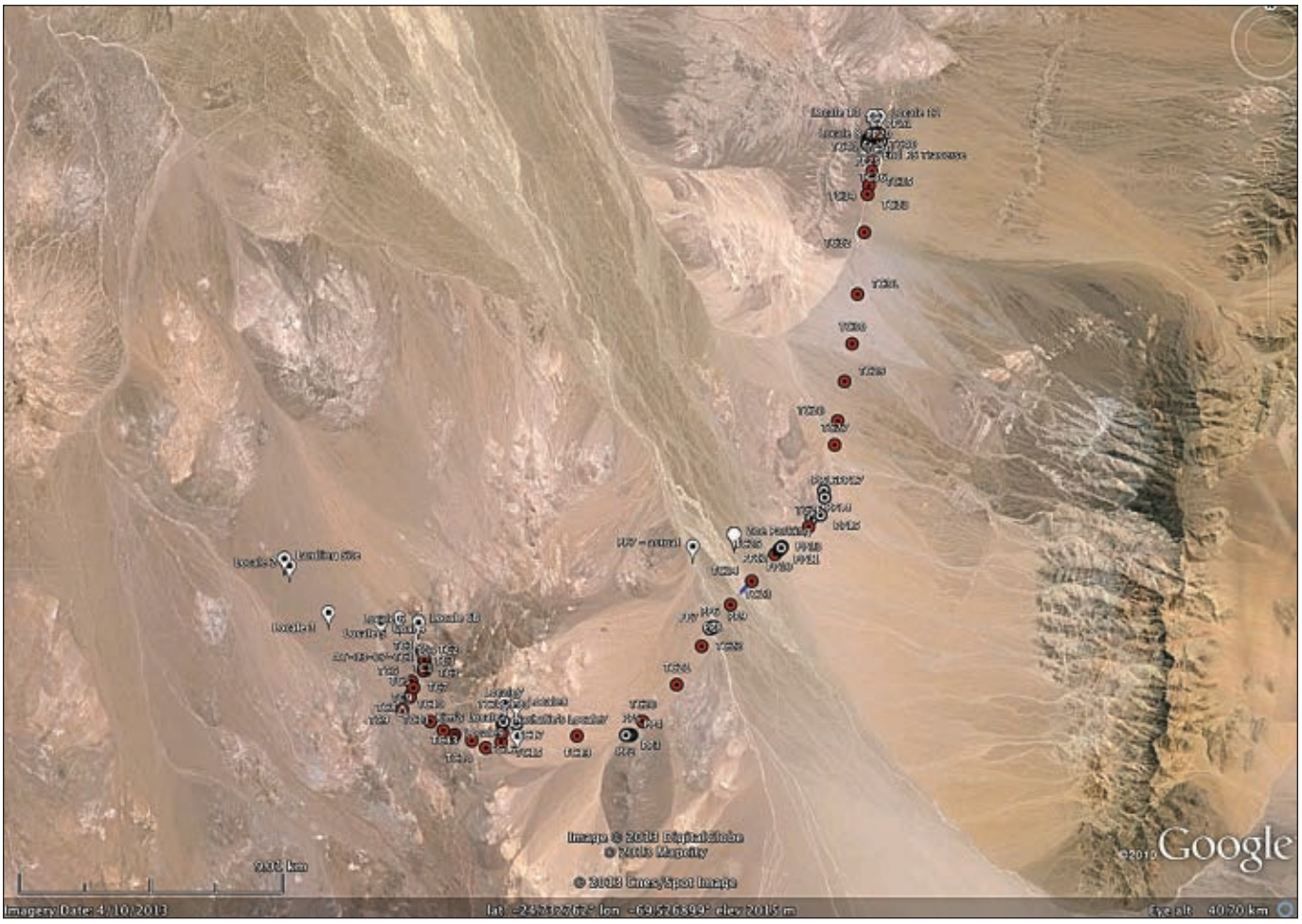


Figure 2. Locales Visited During the 2013 LITA Field Season.

A robot could even divert its path slightly to pursue science targets of opportunity (Woods et al. 2009). Multiday plans could therefore make very efficient use of communications and personnel resources, enhancing long-distance survey missions.

The Life in the Atacama project is a NASA-sponsored effort to evaluate these techniques in the context of desert subsurface biogeology (Cabrol et al. 2007). It uses Zoë (Wettergreen et al. 2008), a rover capable of traveling more than 10 kilometers per day and autonomously drilling up to 0.7 meter depth (figure 1). As a mobility platform it combines navigational autonomy with a changing payload of onboard science instruments. Previous investigations have used a fluorescence imager capable of detecting very specific organic compounds and neutron detectors to measure hydrogen abundance. The current configuration incorporates a Raman spectrometer, a visible near infrared point spectrometer, and navigation and science cameras. During a series of experiments in the summer of 2013, scientists guided Zoë remotely through the desert while exploring its geology and biology.

This article describes the science autonomy system developed and tested with Zoë. It performed automatic acquisition of visible/near infrared (Vis-NIR) reflectance spectroscopy throughout the 2013 field season. This involved a range of different autonomous decisions exercised at various spatiotemporal scales. We begin by describing the rover platform and instrument payload. We then discuss instrument self-calibration, science feature detection, and targeting capabilities. We describe larger-scale path planning used to select informative paths between waypoints. We also detail the operational protocols used to command the rover and the results of its autonomous data collection. These experiments provide a case study of science autonomy deployed continuously over long distances. We report on system performance, lessons learned, and plans for future development.

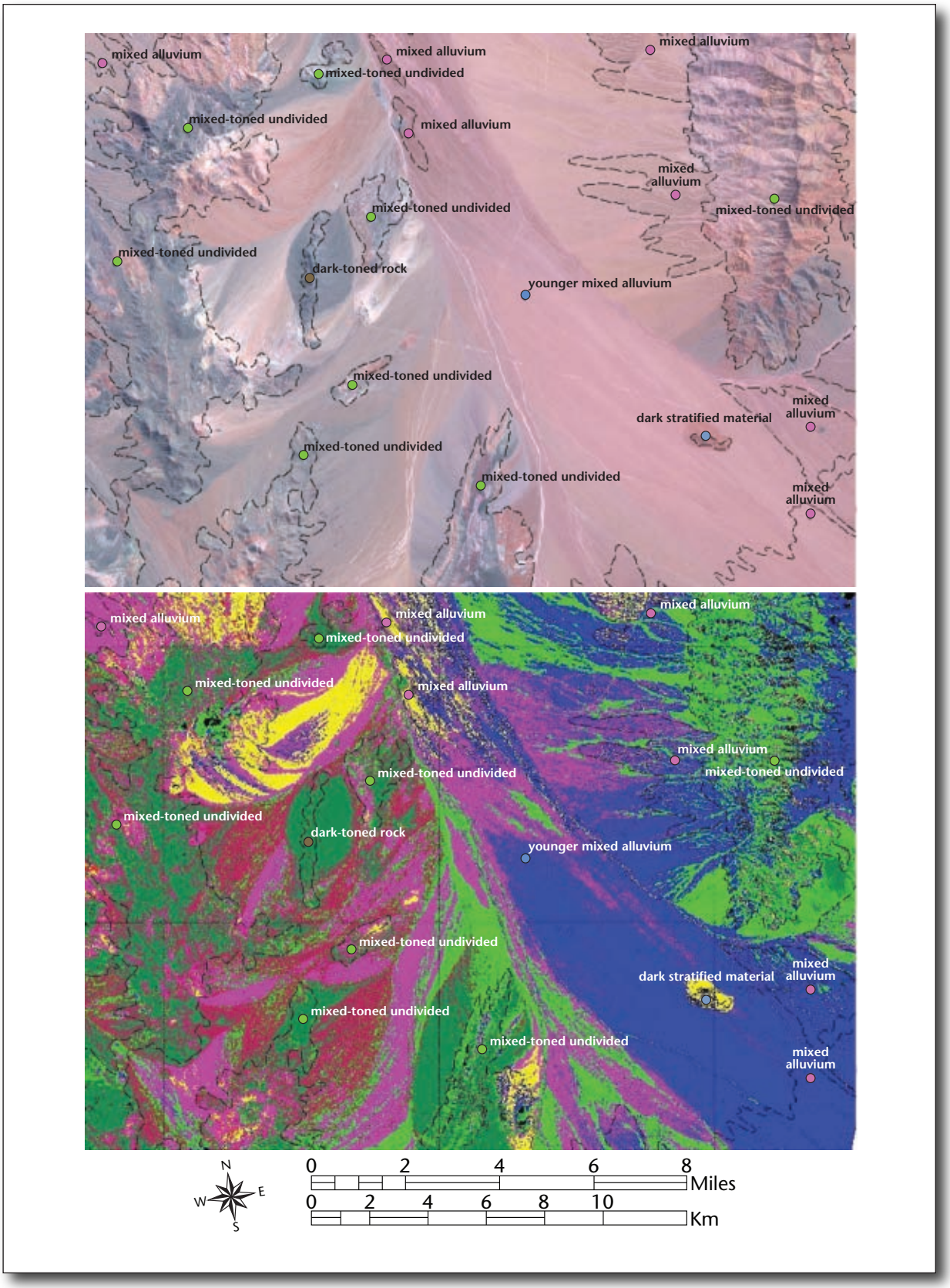


Figure 3. Examples of the Different Data Products.

Top: Landsat image used in traverse planning. Bottom: Geologic classification map derived from ASTER data.

Figure 4. Hazard Avoidance.

Rover Platform, Instruments, and Operations

In typical field work, rover operations follow a daily cycle in which a remote science team reviews the prior data, decides the next day's navigation waypoints and measurements, and then sends these commands to the rover over a satellite link. This is similar to the sporadic communications of a planetary mission. The rover then executes its commands over the course of the subsequent day. In the Atacama campaign, typical command cycles for Zoë cover 5–10 kilometers per day. Figure 2 shows the entire traverse path: red dots show locations for imaging and spectral data collection, while white paddles indicate sites of particular interest where more in-depth study is performed.

Scientists determine the waypoints for the next day using geologic and compositional maps produced from orbital remote sensing data. Here the ASTER instrument proves particularly useful: its images have a spatial resolution of 15 meters (visible) and 30 meters (SWIR), making them capable of resolving details such as isolated rock outcrops. While the three visible and six SWIR bands are not sufficient to conclusively identify mineralogical composition, they do help discriminate the principal units of surface material and suggested representative sites to visit.

Figure 3 shows examples of the different data products: a Landsat image with three visible bands reveals terrain morphology and desirable outcrops, and a multiband ASTER image provides a rough classification of mineralogical units.

The rover itself is capable of driving more than 10 kilometers per day on challenging desert terrain (Wettergreen et al. 2008). On-board obstacle avoidance uses three-dimensional geometry from stereo imagery to identify hazards above the ground plane and plan local drive arcs that go around them (figure 4). Figure 5 shows the robot and the components used by its science autonomy system. A pair of forward-facing navigation cameras provide hazard avoidance capability through a local path planner. The vertical drill structure delivers subsurface soil to a microscopic imager and a Raman spectrometer inside the rover. Analyzing the drill samples takes an hour or more, so these are deployed judiciously at specific locations. However, we found that autono-

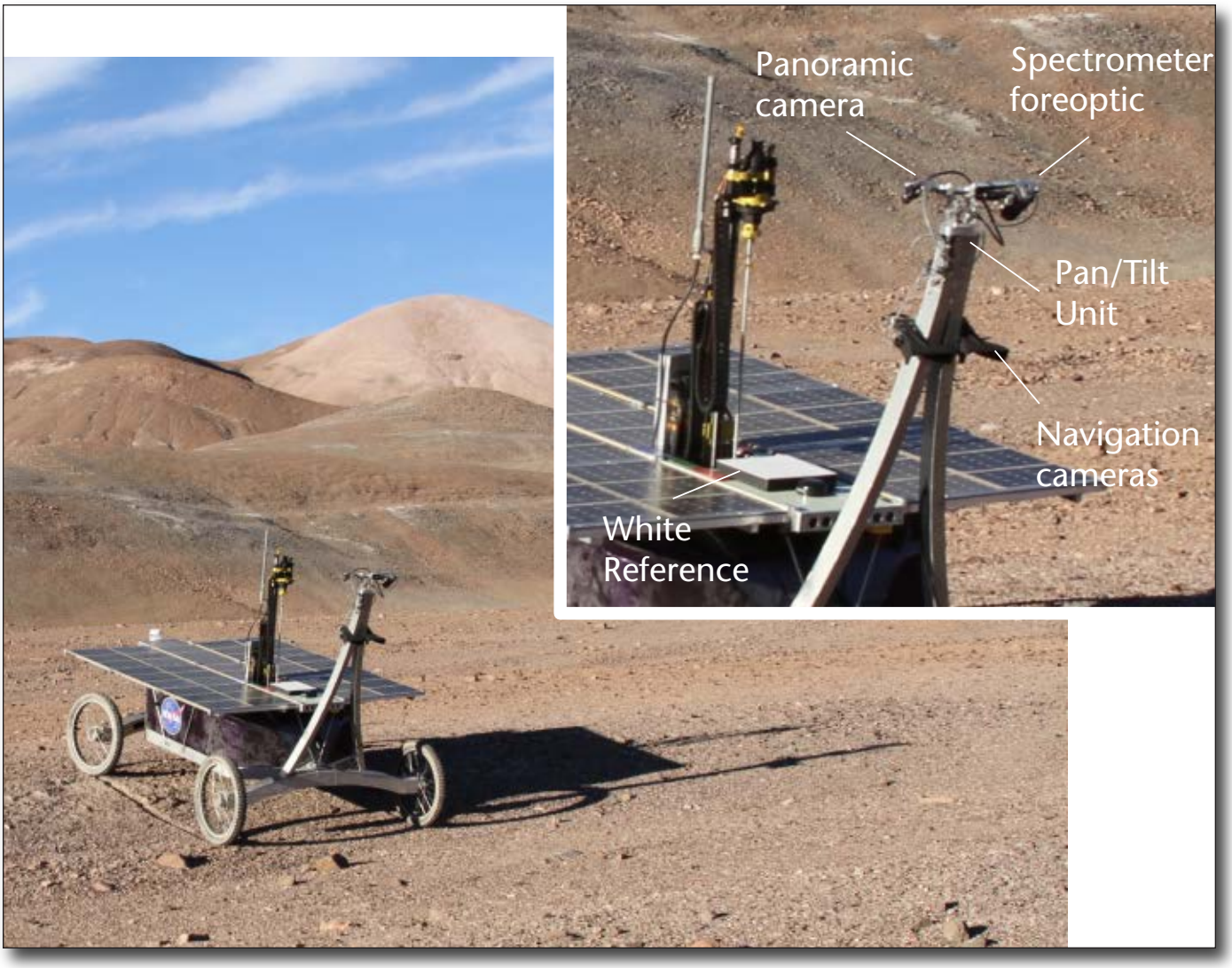


Figure 5. The Main Components of Zoë's Vis-NIR Spectrometer System.

A Raman spectrometer inside the rover measures pulverized samples from the subsurface drill.

my could play a role in improving the science data collected by the Vis-NIR spectrometer. The spectrometer is a modified Analytical Spectral Devices Fieldspec Pro that acquires radiance spectra from 0.4–2.5 micrometers at 0.001 micrometer resolution, housed in the rover body and connected by a fiber optic cable to a foreoptic telescope mounted on a pan-tilt mechanism. The foreoptic provides a 1 degree field of view, and can be directed at specific targets in the environment. Its field of regard spans a full 360 degrees azimuth and 90 degrees elevation. A colocated camera provides visual context to interpret the spectra.

Zoë's Vis-NIR reflectance data overlaps in wavelength with ASTER orbital images; it is a more spatially and spectrally refined version of the satellite data. By visiting distinctive terrain units of figure 3, analysts can refine the remote view with detailed spectral information and specific mineral absorption features. In this manner the Vis-NIR data serves as both a validation of the orbital data and a means to better interpret the mineralogical constraints and context for biogeology studies. Each session of Vis-NIR acquisitions begins with the rover calibrating its instrument for temperature, solar geometry, and atmospheric conditions using a white reference target mounted on the rover deck (figure 5 inset).

Dividing the radiance from the target by the reference measurement produces reflectance data of the form shown in figure 6. These spectra were acquired at locations indicated in the adjacent panoramic camera subframe, from a distance of approximately 2 meters. The reflectance values represent the fraction of light reflected at each wavelength; more specific

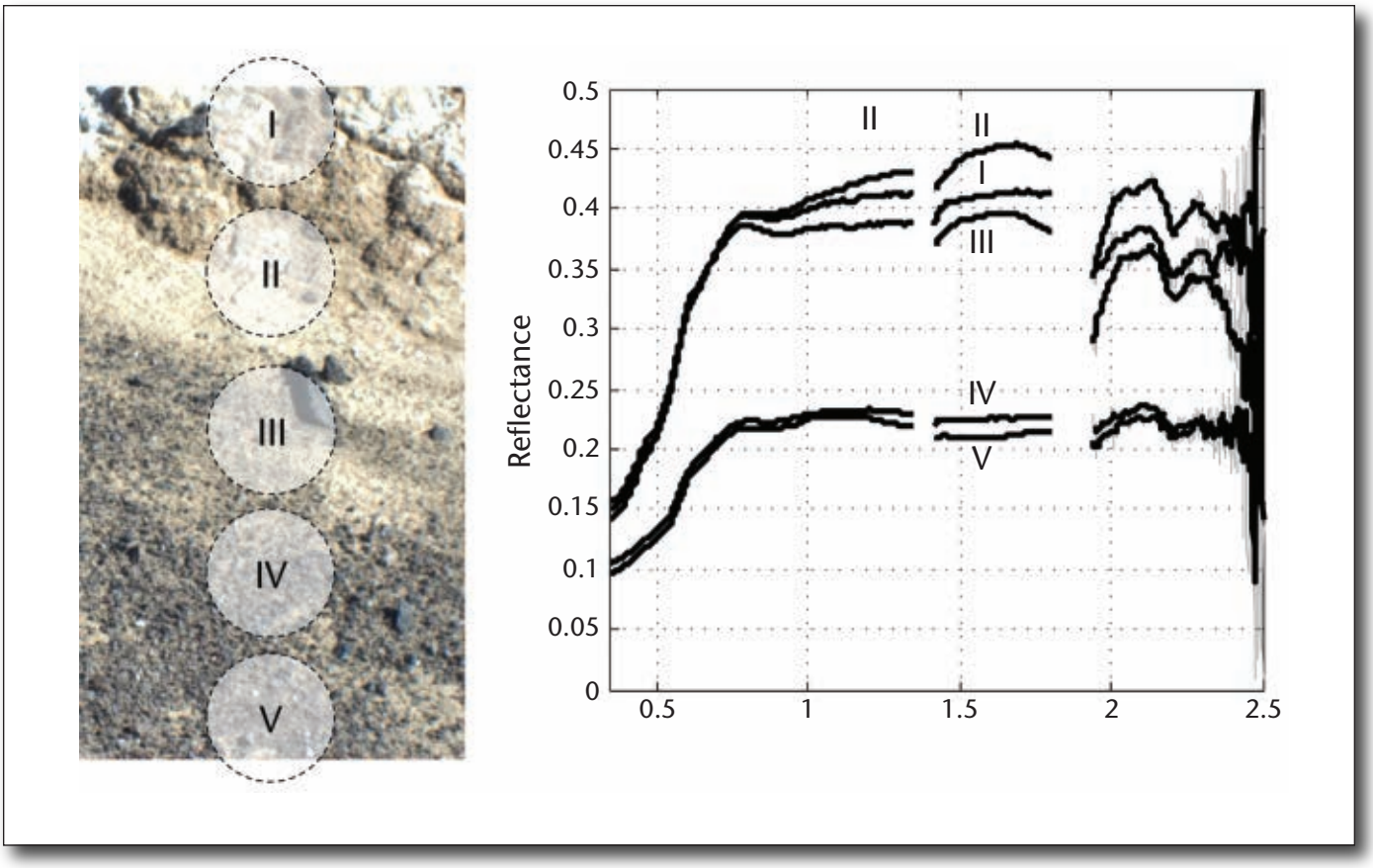


Figure 6. Panoramic Camera Subframe.

Dive spectrometer fields of view (PP24), and associated reflectance spectra.

formulations are possible (Schaepman-Strub et al. 2006), but we will use reflectance here in the ordinary Lambertian sense. This assumption should generally hold for the geologic materials of interest. Note that the light-colored sediment area in spectra I-III is associated with a higher average reflectance, as well as unique spectral features such as the dip near 2 micrometers. These spectra were smoothed using local linear regression, but some lingering noise spikes at longer wavelengths evidence the lower signal level in these spectral regions.

Science Autonomy Methods

Zoë's science autonomy system includes two basic capabilities that operate on mesoscale and macroscale features respectively. Smart targeting can identify science features in rover navigation imagery and use this information to point the Vis-NIR spectrometer. Adaptive path planning navigates on scales of tens or hundreds of meters, using satellite images to select waypoints with distinctive or novel spectra. We describe each of these techniques in turn.

Smart Targeting

Zoë began each autonomous target selection process by acquiring a navigation camera image. On-board image processing then analyzed the scene to find large contiguous regions of a desired terrain class. Typically these classes were rough surface features like rock outcrop or bright sediment patches with distinctive spectral signatures. Upon finding a feasible target, the rover recalibrated its Vis-NIR spectrometer, pointed at the feature, and collected a small 3 x 3 raster of spectra centered on the target of interest. For context, it also acquired a high-resolution color image of the scene.

The image analysis used a random forest pixel classification system described in previous work (Foil et al. 2013; Wagstaff et al. 2013) and adapted to the Atacama environment. This supervised classification method learns a mapping from local pixel intensities to the surface class of that pixel. The model is instantiated as an ensemble of decision trees trained in advance. At run time, the rover tested each pixel in the new image and averaged the classification of each tree in the ensemble. The end result was a classification map of the entire image, along with asso-

ciated class posterior probabilities. By subsampling each image by a factor of four prior to classification, processing time was less than a second on Zoë's onboard laptop-scale CPU.

After image classification, connected components analysis was used to identify contiguous targets. The rover then promoted the single largest target of the desired class for followup data collection. For each target, a center pixel was determined using the largest inscribed circle heuristic (Estlin et al. 2012) and transformed to a pan-tilt angle using the assumption of a planar terrain surface. Use of navigation camera stereo data would identify a true three-dimensional position and enable more sophisticated kinematic solutions. Here we relied on an approximate planar solution coupled with rastering to ensure that an inaccurate pointing would still capture the target in at least one of the Vis-NIR spectra.

Scientists developed several different ways to incorporate this capability into multiday rover operations. The first approach was a target check used in the middle of long traverses. This only deployed the spectrometer if a feature of interest was found at the check point. If there was no feature in the rover's field of view, it would carry on without spending the time to calibrate and deploy its spectrometer. In this fashion, Zoë could cover long distances without spending undue time on bare or uninteresting terrain. This strategy was also useful near the boundary of geologic contacts where the precise location was uncertain. A second strategy involved a paired panorama that acted as a supplement to a commanded Vis-NIR spectral raster. Here the rover committed all time resources in advance. It calibrated its spectrometer and acquired data in columns of five spectra spaced at 10 degree increments directly in front of the robot and to either side. This provided a representative sampling of the terrain comprising the rover's current ASTER pixel. It then augmented this dataset with a 3 x 3 raster centered on any target of interest. Together, these two products gave a better insight than either taken individually. They met the dual needs of having representative spectra as well as capturing distinctive (outlier) features.

Adaptive Path Planning

The science autonomy system also operates on larger scales of tens or hundreds of meters, where it analyzes satellite data to adjust its traverse path. We model the explored environment using a standard geographic or area mixing model where each measurement is a mixture of a small number of end-member materials. End members' spectra combine in proportion to their physical extent on the surface. Most scenes contain just a few end-member spectra, and any measurement $\mathbf{x}$ can be reconstructed with appropriate constituents and mixing fractions. For a scene with m end members we define the mixing fractions to be vectors $\phi \in \mathbb{R}^m$. More generally we can model a spectral image using a linear combination of library spectra given by a $d \times m$ matrix Y. This gives the relationship $x = Y\phi$.

In practice there is always residual error separating the reconstruction from the measurement. This is partly attributable to measurement noise, but unless the library is comprehensive there may also be incompleteness errors (for example, spectral features that are expressed in the observations but not present in the library). A library that reconstructs all spectra well can be said to have explained the scene, and provides insight into the mineral compositions in the remote sensing data. This intuition provides a figure of merit for an adaptive path-planning system to select future measurement locations. Zoë's planner selects locations, the measurements at which, when used to augment the collected library Y, provide the largest expected reduction in unmixing error. The planner aims to visit locations that are spectrally distinctive, collecting samples that fully explain the orbital image.

In detail, we begin with a space of candidate measurement locations $\mathcal{L}$. The robot collects a library of spectra by sampling at sites $B = \{b : b \in \mathcal{L}\}$. We define a stochastic measurement function, $\mathbf{y} = f(b) + \varepsilon$ with Gaussian-distributed measurement noise ε, that yields spectral measurements $Y = \{\mathbf{y}_i : \mathbf{y}_i \in \mathbb{R}^d, 1 \le i \le m\}$. Together the observations form a spectral library, a random $m \times d$ matrix written Y_B. Good measurements reduce the total reconstruction error for selected remote sensing observations given by $X = \{\mathbf{x}_i : \mathbf{x}_i \in \mathbb{R}^d, 1 \le i \le n\}$. We impose a resource cost $C(B)$ to represent limited time, power, and bandwidth; it must not exceed a total budget β. For simplicity we will initially ignore the cost of point-to-point travel.

We define a risk function as the expected reconstruction error incurred from unmixing the remote images with the library of spectra collected at the surface:

$$R(B) = E\left[\min_{\phi} \sum_{\mathbf{x} \in X} \left\| Y_B\phi - \mathbf{x} \right\|_2\right] \quad \text{for } \phi \ge 0,\ C(B) \le \beta \tag{1}$$

Here we are taking the expectation over the rover's observation matrix, which is a random variable. Computing this expectation is analytically challenging, so instead we solve the related problem:

$$\arg\min_B = \min_{\phi} \sum_{\mathbf{x} \in X} \left\| X_B\phi - \mathbf{x} \right\|_2 \quad \text{for } \phi \ge 0, C(B) \le \beta \tag{2}$$

The linear geographic mixing assumption allows us to infer the expectation $E[YB]$ since in situ spectra combine in proportion to their extent on the surface. Due to geographic mixing we can directly substitute the remotely observed spectra as the expectated observations. We rewrite the objective function using remote measurements at sites B, written X_B:

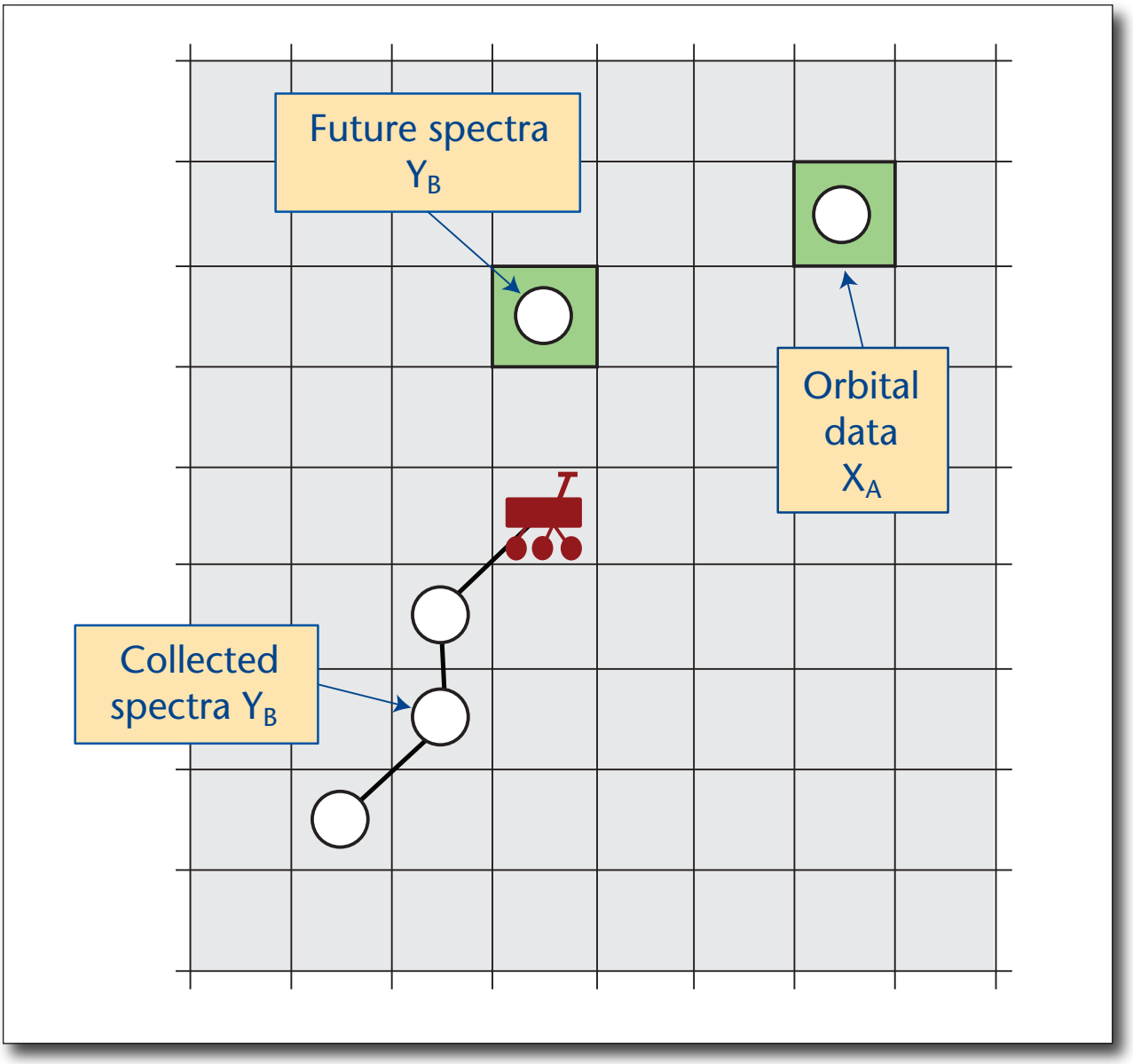


Figure 7. Formulation of Adaptive Path Planning.

$$R(B|A) = \min_{\phi} \sum_{x \in X} \|X_B \phi - x\|_2 \quad \text{for } \phi \geq 0, C(B) \leq \beta \tag{3}$$

This allows direct computation of the objective for any candidate set of measurement locations.

As the robot begins to collect spectra, some elements of $E[Y]$ become observed. The matrix Z_A represents the library of spectra collected at previous locations $A = \{a : a \in \mathcal{L}.\}$. These measurements are a realization of Y_A, and can be substituted into the expectation as the library of in situ spectra grows. Consequently the library used for unmixing consists of (1) the actual spectra collected at previous locations concatenated with (2) the expected spectra collected at the future candidate locations. The objective is:

$$R(B|A) = \min_{\phi} \sum_{x \in X} \|[Z_A X_B]\phi - x\|_2 \quad \text{for } \phi \geq 0, C(B) + C(A) \leq \beta \tag{4}$$

To summarize, this final form reflects the key elements of Vis-NIR surface exploration: the overall goal of an accurate model using a handful of spectra, reflected in the squared error term; the physical behavior of geographic mixing, which appears as a positivity constraint; and the overall path-length budget β representing finite rover energy and time resources.

Figure 7 portrays the planning process. Here the robot has collected two spectra to form its library Z_A.

It calculates the expected risk of a candidate path using remote sensing data at locations X_B as a proxy for future measurements. In this manner, it can greedily (or nonmyopically) construct an optimal path. For our tests, the path planning was purely greedy; we added waypoints one by one, inserting them into the optimal location in the waypoint sequence and stopping when the total Euclidean path cost was exceeded.

During the Atacama field season we defined a budget defined in terms of path length, typically providing 1.5 times the straight-line distance to the goal. The time cost could be significant for longer traverses, particularly including the cost of the spectral measurements at intermediate waypoint. For this reason, most navigation actions were driving commands with adaptive navigation actions at particular regions of special interest. When it encounters a science waypoint in the plan, the science path-planning software finds the complete interpolating path that minimizes spectral reconstruction error of the nine-band ASTER image. It drives to the next intermediate waypoint along that path, collects a spectrum of the terrain, and replans the remainder using whatever path budget remains. That remainder becomes the next science plan, which is further refined in additional planning rounds as the rover progresses forward. In this fashion, the science planner can be fully stateless and react to new data encountered during the traverse.

Figure 8 shows the benefit of science-aware path planning in a simple simulation. We simulate a virtual rover traversing the famous Cuprite, Nevada, mining district, which is known for containing many distinctive spectral features of interest in highly localized outcrops (Swayze et al. 1992). Here we planned rover paths using orbital ASTER data, simulating 256 trials with random start and end points within the scene. We also simulated high-resolution in situ acquisitions using coregistered data from the Airborne Visible Near Infrared Spectrometer (AVIRIS) (Green et al. 1998).

The comparison considers four different strategies to fill the path-length budget: a random path, which bends the path toward randomly selected intermediate waypoints; a direct path, which begins with a straight line and then adds "wiggles" until the total length is reached; an unconstrained adaptive approach that minimizes equation 3 but without the positivity constraint; and an adaptive approach that enforces positivity of mixing fractions. We recomputed the reconstruction error for every trial by applying nonnegative least squares to the collected high-resolution spectra. Figure 8 shows each method's performance with a box indicating the median and quartile of the data and the whiskers indicating the extrema. Both adaptive methods significantly outperform the nonadaptive approaches, with the constrained adaptive method performing best of all. This performance boost happens because only the adaptive systems actively pursue the isolated outcrops with unique mineralogy.

On-board the real rover, a low-level control system is required to travel safely between these features of interest. Consequently, Zoë has on-board navigation software to turn high-level science waypoints, spaced on the order of tens or hundreds of meters, into low-level vehicle actions like drive arcs. It uses a software suite known as the Reliable Autonomous Surface Mobility (RASM) package (Wettergreen and Wagner 2012) capable of local hazard avoidance and path planning using a three-dimensional terrain representation. RASM extracts a cloud of three-dimensional points from the stereo cameras, orients these points relative to previously collected data, and builds a triangulated mesh. An A* search algorithm projects drive arcs across this mesh to compute the cost of local control actions. On longer scales, graph search identifies the best path to the next waypoint. RASM retains knowledge of the topology relating observation locations to their neighbors, permitting efficient loop closure and pose estimation over long distances.

Field Season Results

We engaged smart targeting during three days of rover operations. Table 1 shows the performance for typical targets during these traverses. Columns indicate the day (Sol) of operations; the action sequence number, with TC indicating a target check, PP a paired panorama, and AT a more specific planned data collection activity; the analysts' post hoc interpretation of the feature that was found; and two columns indicating whether the result was a reasonable science target and whether the pointing was accurate. Pointing accuracy was evaluated based on the context image collected with each spectrum, allowing it to be placed within the navigation camera image.

Overall, target selection performed reliably. The majority of targets were either rocks or patches of distinctive sediment. The only arguable failure was when the system identified a distant car that fell into the rover field of view. The pointing solution was slightly less reliable, since our groundplane assumption tended to break down at the periphery of the navigation camera image near the horizon.

Occasionally very distant targets would result in the rover aiming its spectrometer too far into the distance. Only one scene was totally featureless — an empty plain — and in this case the rover detected no targets and successfully abstained from spending any time resources.

Figure 9 shows several images from the real-time classification. Here the system was trained to recognize rocks and high albedo soil patches, and it successfully finds these features. In the center column, a red overlay represents the surface labeled as the belonging to the target class. The black rectangles

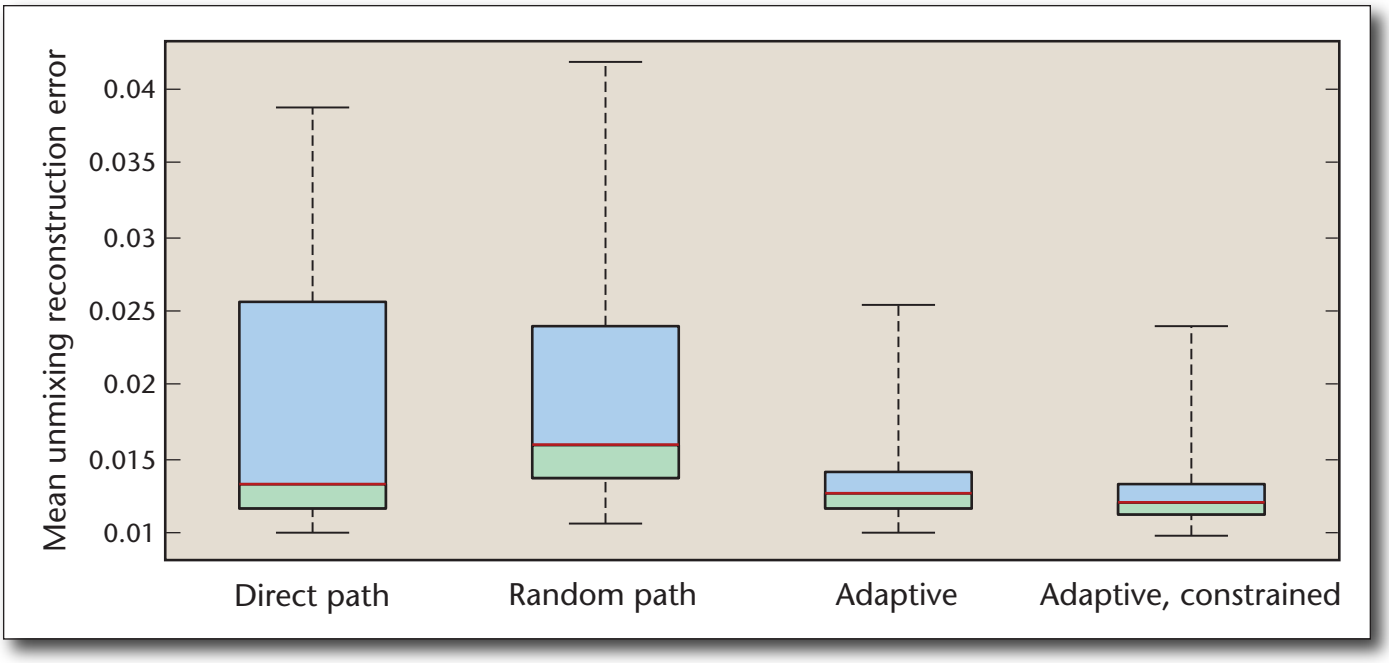


Figure 8. Adaptive Path-Planning Performance, in Simulation.

Sol	Action	SR	Target Found	Target Valid?	Pointing Accurate?	Notes
12	TC30	122	Rocks	OK	OK	
	TC31a	123	Foreground rock	OK	OK	
	TC31b	125	Foreground rock	OK	OK	
	TC32	128	Rock pile, sediment	OK	OK	
13	AT-13-09	172	Disturbed rocks and sediment	OK	OK	
	PP22	166	Distant rock patch	OK	BAD	1
	PP23	160	Distant rock patch	OK	OK	
	PP24	154	Foreground rocks	OK	OK	
	PP25	148	Foreground rocks	OK	OK	
	TC34	158	Foreground sediment patch / rocks	OK	OK	
	TC34-recon	132	None	OK	n/a	2
	TC41	170	Rocks	OK	BAD	3
	TC42	164	Distant rock patch	OK	BAD	4
14	AT-13-10	190	Car	BAD	BAD	5
	PP19	216	Foreground Rock	OK	OK	
	TC40	183	Rock patch	OK	OK	

Table 1. Target Detection Results from Playa Exploration Phase.

1: Aims too high for distant targets. 2: No target in scene. 3: Targeted feature not the largest rock. 4: Very distant feature. 5: Cars in frame.

show the field of view of the high-resolution follow-up image collected by the mast-mounted camera (right column). Each follow-up image is accompanied by a 3 x 3 raster of spectra. Even when the target selection was successful, we did not notice a significant difference between the on- and off-target spectra. This may be attributed to low signal to noise. Alternatively, these features may have spectral signatures that were very similar to the background substrate.

We deployed the adaptive navigation system successfully in two instances during the 2013 field campaign. Near the end of the field season the rover vis-

Figure 9. Examples of Targets Detected in Midtraverse, and Associated Followup Images.

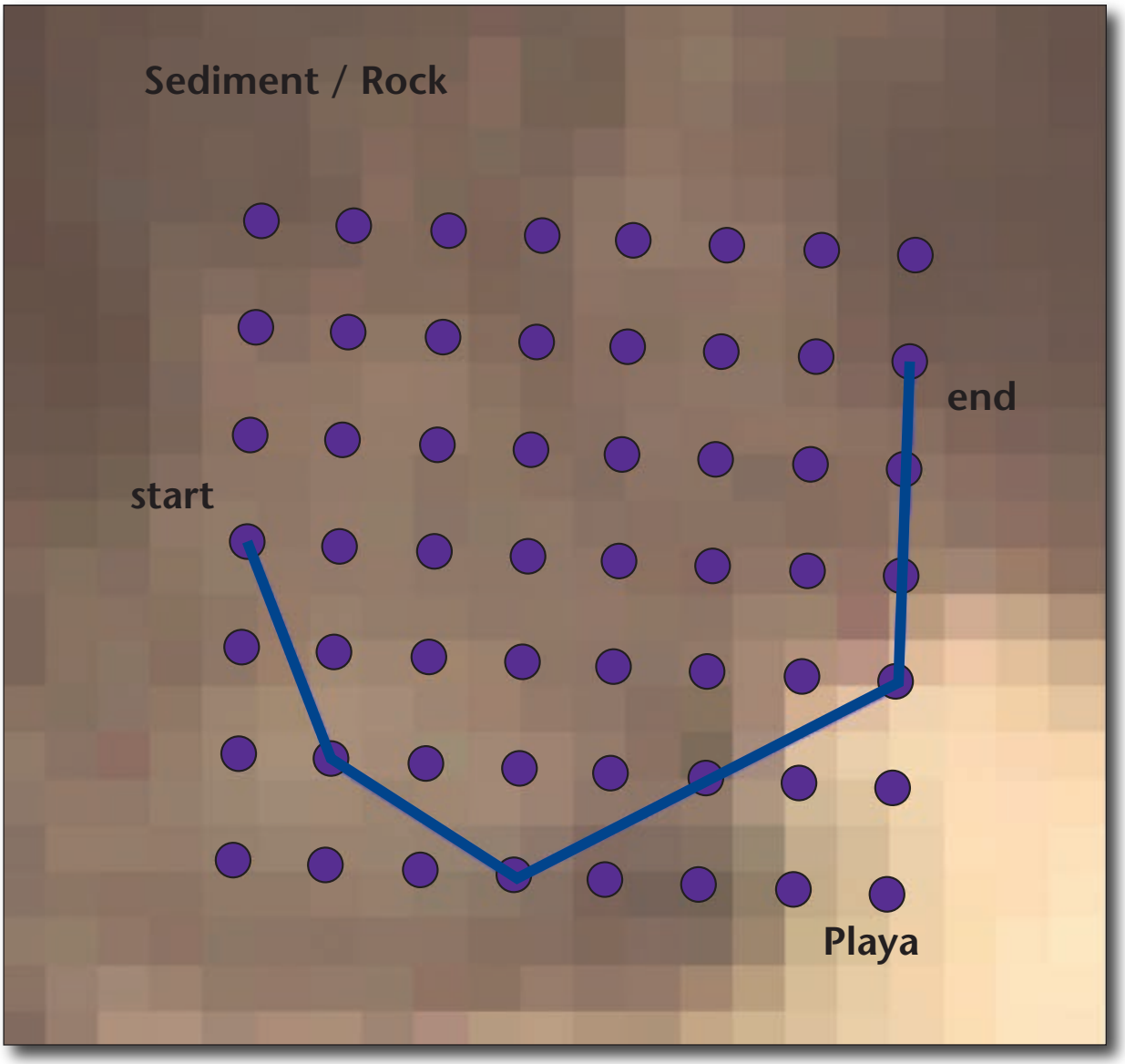


Figure 10. A Composite of Rover Images Showing the Playa Where Adaptive Navigation Was Evaluated.

ited a playa — a dry lakebed approximately 2 kilometers in length that was spectrally distinctive from the surrounding terrain (figure 10). Figure 11 shows a typical round of adaptive path planning near the playa edge. Here the playa is visible as a bright area in the lower right of the overhead satellite image. The large pixels represent 15-meter ASTER data. Here Zoë's planned path, in blue, diverts to sample the spectrally distinctive playa surface. The path changed only slightly in subsequent replanning as the rover visited each waypoint and incorporated the new spectra into its solution.

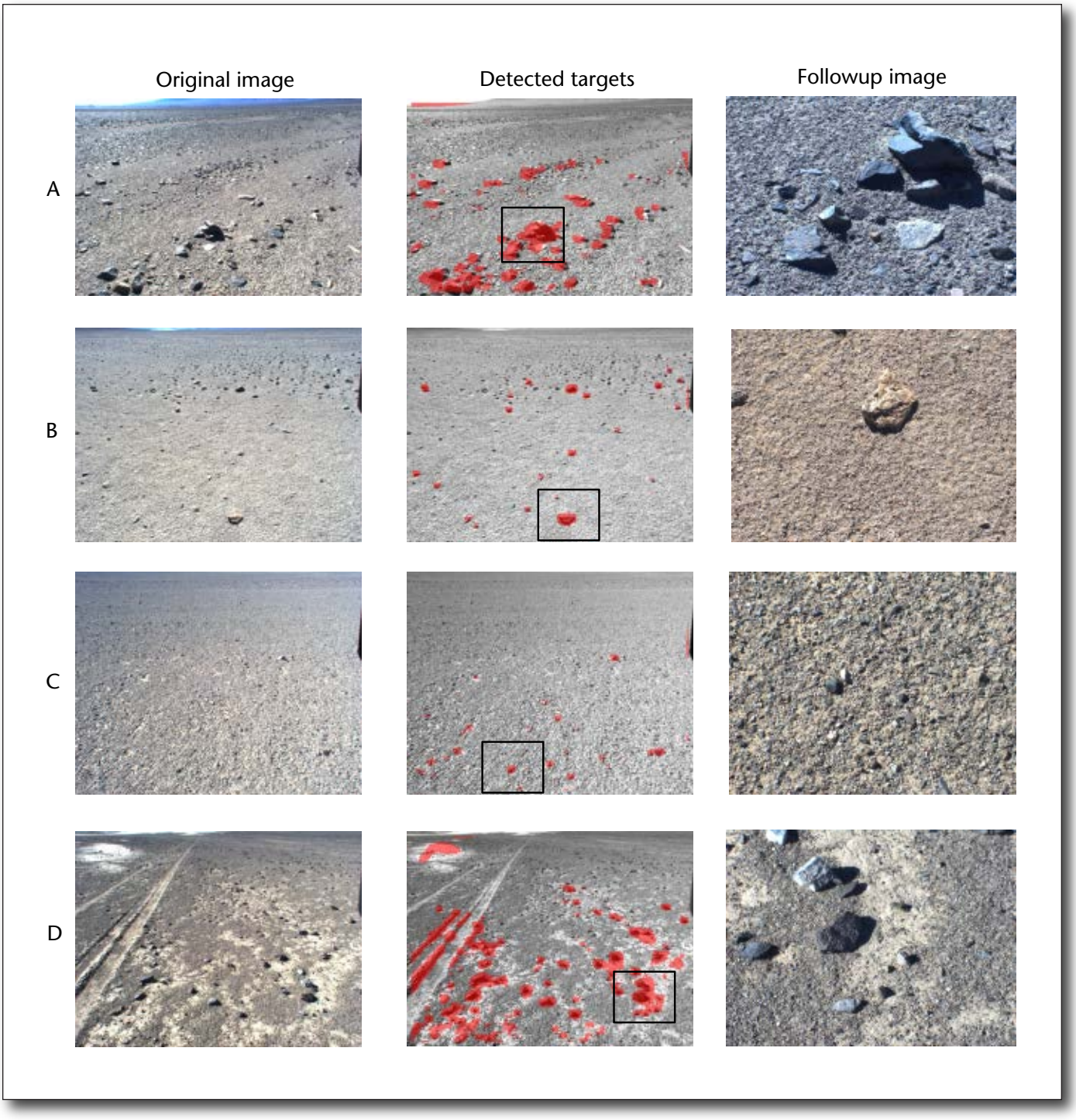


Figure 11. Demonstration of Adaptive Path Planning.

While the on-board planning gave an intuitive and reasonable answer, the actual rover paths were not as expected due to misregistration between orbital data products and the rover's on-board GPS estimate. Postanalysis of the data revealed the real position was offset by more than 100 meters from the intended location, so the actual rover path spent most of its time on the playa. In the future we will directly address these registration errors with the use of explicit ground control points (GCPs).

Conclusions

This work demonstrates novel techniques integrating adaptive autonomous science activities with pre-planned data collection. Zoë's system will continue to mature in the coming year.

Acknowledgments

A portion of this research was performed at the Jet Propulsion Laboratory, California Institute of Technology. The Landsat map in Figure 3 is courtesy Ken Tanaka (USGS). The geologic classification comes from Jeffery Moersch (UTenn). Copyright 2013. This research is funded by the NASA Astrobiology and Technology for Exploring Planets (ASTEP) program under grant NNX11AJ87G.

References

Cabrol, N. A.; Wettergreen, D.; Warren-Rhodes, K.; Grin, E. A.; Moersch, J.; Diaz, G. C.; Cockell, C. S.; Coppin, P.; Demergasso, C.; Dohm, J. M.; Ernst, L.; Fisher, G.; Glasgow, J.; Hardgrove, C.; Hock, A. N.; Jonak, D.; Marinangeli, L.; Minkley, E.; Ori, G. G.; Piatek, J.; Pudenz, E.; Smith, T.; Stubbs, K.; Thomas, G.; Thompson, D.; Waggoner, A.; Wagner, M.; Weinstein, S.; and Wyatt, M. 2007. Life in the Atacama: Searching for Life With Rovers (Science Overview). *Journal of Geophysical Research: Biogeosciences* 112(G4): 28, 2007. dx.doi.org/10.1029/2006JG000298

Castaño, A.; Fukunaga, A.; Biesiadecki, J.; Neakrase, L.; Whelley, P.; Greeley, R.; Lemmon, M.; Castano, R.; and Chien, S. 2008. Automatic Detection of Dust Devils and Clouds on Mars. *Machine Vision and Applications* 19(5-6): 467–482. dx.doi.org/10.1007/s00138-007-0081-3

Chien, S.; Sherwood, R.; Tran, D.; Cichy, B.; Rabideau, G.; Castano, R.; Davies, A.; Mandl, D.; Frye, S.; Trout, B.; Shulman, S.; and Boyer, D. 2005. Using Autonomy Flight Software to Improve Science Return on Earth Observing One. *Journal of Aerospace Computing, Information, and Communication* 2(4): 196–216. dx.doi.org/10.2514/1.12923

Davies, A.; Chien, S.; Baker, V.; Doggett, T.; Dohm, J.; Greeley, R.; Ip, F.; Cichy, B.; Rabideau, G.; Tran, D.; Sherwood, R. 2006. Monitoring Active Volcanism With the Autonomous Sciencecraft Experiment on EO-1. *Remote Sensing of Environment* 101(4): 427–446. dx.doi.org/10.1016/j.rse.2005.08.007

Doggett, T.; Greeley, R.; Chien, S.; Castano, R.; Cichy, B.; Davies, A.; Rabideau, G.; Sherwood, R.; Tran, D.; Baker, V.; et al. 2006. Autonomous Detection of Cryospheric Change with Hyperion On-Board Earth Observing-1. *Remote Sensing of Environment* 101(4): 447–462. dx.doi.org/10.1016/j.rse.2005.11.014

Estlin, T.; Bornstein, B.; Gaines, D.; Anderson, R. C.; Thompson, D. R.; Burl, M.; Castano, R.; and Judd, M. 2012. Aegis Automated Targeting for Mer Opportunity Rover. *ACM Transactions on Intelligent Systems Technology* (TIST) 3(3): Article 50. dx.doi.org/10.1145/2168752.2168764.

Foil, G.; Thompson, D. R.; Abbey, W.; and Wettergreen, D. S. 2013. Probabilistic Surface Classification for Rover Instrument Targeting. In *Proceedings of the 2013 IEEE International Conference on Intelligent Robots and Systems* (IROS), 775–782. Piscataway, NJ: Institute of Electrical and Electronics Engineers. dx.doi.org/10.1109/IROS.2013.6696439

Green, R. O.; Eastwood, M. L.; Sarture, C. M.; Chrien, T. G.; Aronsson, M.; Chippendale, B. J.; Faust, J. A.; Pavri, B. E.; Chovit, C. J.; Solis, M.; Olah, M. R.; Williams, O. 1998. Imaging Spectroscopy and the Airborne Visible/Infrared Imaging Spectrometer (Aviris). *Remote Sensing of Environment* 65(3): 227–248. dx.doi.org/10.1016/S0034-4257(98)00064-9

Ip, F.; Dohm, J.; Baker, V.; Doggett, T.; Davies, A.; Castano, R.; Chien, S.; Cichy, B.; Greeley, R.; Sherwood, R.; Tran, D.; Rabideau, G. 2006. Flood Detection and Monitoring with the Autonomous Sciencecraft Experiment Onboard EO-1. *Remote Sensing of Environment* 101(4): 463–481. dx.doi.org/10.1016/j.rse.2005.12.018

Mastrodemos, N.; Kubitschek, D. G.; and Synnott, S. P. 2005. Autonomous Navigation for the Deep Impact Mission Encounter with Comet Tempel 1. *Space Science Reviews* 117(1–2): 95–121. dx.doi.org/10.1007/s11214-005-3394-4

Schaepman-Strub, G.; Schaepman, M.; Painter, T.; Dangel, S.; and Martonchik, J. 2006. Reflectance Quantities in Optical Remote Sensing: Definitions and Case Studies. *Remote Sensing of Environment* 103(1): 27–42. dx.doi.org/10.1016/j.rse.2006.03.002

Swayze, G.; Clark, R.; Sutley, S.; and Gallagher, A. 1992. *Ground-Truthing Aviris Mineral Mapping at Cuprite, Nevada. Summaries of the Third Annual JPL Airborne Geosciences Workshop, Volume 1: AVIRIS Workshop*. JPL Publication 92-14, 47–49. Pasadena, CA: Jet Propulsion Laboratory, California Institute of Technology.

Thompson, D.; Bornstein, B.; Chien, S.; Schaffer, S.; Tran, D.; Bue, B.; Castano, R.; Gleeson, D.; and Noell, A. 2013. Autonomous Spectral Discovery and Mapping Onboard the EO-1 Spacecraft. *IEEE Transactions on Geoscience and Remote Sensing* 51(6): 3567–3579. dx.doi.org/10.1109/TGRS.2012.2226040

Wagstaff, K.; Thompson, D. R.; Abbey, W.; Allwood, A.; Bekker, D. L.; Cabrol, N. A.; Fuchs, T.; and Ortega, K. 2013. Smart, Texture-Sensitive Instrument Classification for In Situ Rock and Layer Analysis. *Geophysical Research Letters* 40(16): 4188–4193. dx.doi.org/10.1002/grl.50817

Wettergreen, D., and Wagner, M. 2012. Developing a Framework for Reliable Autonomous Surface Mobility. Paper presented at the 12th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS-12), Turin, Italy, 4–6 September.

Wettergreen, D.; Wagner, M. D.; Jonak, D.; Baskaran, V.; Deans, M.; Heys, S.; Pane, D.; Smith, T.; Teza, J.; Thompson, D. R.; Tompkins, P.; and Williams, C. 2008. Long-Distance Autonomous Survey and Mapping in the Robotic Investigation of Life in the Atacama Desert. Paper presented at the 9th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS-08), Los Angeles, CA USA, 26–29 February.

Woods, M.; Shaw, A.; Barnes, D.; Price, D.; Long, D.; and Pullan, D. 2009. Autonomous Science for an Exomars Roverlike Mission. *Journal of Field Robotics* 26(4): 358–390. dx.doi.org/10.1002/rob.20289

David Wettergreen is a research professor at the Robotics Institute at Carnegie Mellon University. His research focuses on robotic exploration underwater, on the surface, and in air and space; and in the necessary ingredients of perception, planning, learning and control for robot autonomy. His work spans conceptual and algorithm design through field experimentation and results in mobile robots that explore the difficult, dangerous, and usually dirty places on Earth, in the service of scientific and technical investigations. Much of this work is relevant to ongoing and future space activities.

Greydon Foil is a Ph.D. student at the Robotics Institute at Carnegie Mellon University. He is interested in large-scale mapping and planning for autonomous rovers, with current work involving physical process modeling for better terrain understanding.

Michael Furlong is a researcher with interests in theoretical neuroscience, decision making, and robotics.

David R. Thompson is a researcher at the Jet Propulsion Laboratory, California Institute of Technology. His work involves the application of computer vision, machine learning, and spectroscopy to robotic Earth and planetary science. His algorithms have run on autonomous robots and sensors fielded in North America, South America, the Atlantic Ocean, airborne demonstrations, low Earth orbit, and the surface of Mars.

Multirobot Coordination for Space Exploration

Logan Yliniemi, Adrian K. Agogino, Kagan Tumer

■ *Teams of artificially intelligent planetary rovers have tremendous potential for space exploration, allowing for reduced cost, increased flexibility, and increased reliability. However, having these multiple autonomous devices acting simultaneously leads to a problem of coordination: to achieve the best results, they should work together. This is not a simple task. Due to the large distances and harsh environments, a rover must be able to perform a wide variety of tasks with a wide variety of potential teammates in uncertain and unsafe environments. Directly coding all the necessary rules that can reliably handle all of this coordination and uncertainty is problematic. Instead, this article examines tackling this problem through the use of coordinated reinforcement learning: rather than being programmed what to do, the rovers iteratively learn through trial and error to take take actions that lead to high overall system return. To allow for coordination, yet allow each agent to learn and act independently, we employ state-of-the-art reward-shaping techniques. This article uses visualization techniques to break down complex performance indicators into an accessible form and identifies key future research directions.*

Imagine for a moment that you're tasked with teleoperating (controlling with a joystick) a Mars rover as it navigates across the surface. You watch the feed from the on-board camera as the rover rolls along the surface, when you notice the terrain changing ahead, so you instruct the rover to turn. The problem? You're 6 minutes too late. Due to the speed-of-light delay in communication between yourself and the rover, your monolithic multimillion dollar project is in pieces at the bottom of a Martian canyon, and the nearest repairman is 65 million miles away.

There are, of course, solutions to this type of problem. You can instruct it to travel a very small distance and reevaluate the rover's situation before the next round of travel, but this leads to painfully slow processes that take orders of magnitude longer than they would on Earth. The speed of light is slow enough that it hinders any attempts at interacting regularly with a rover on another planet.

But what if, instead of attempting to control every aspect of the rover's operation, we were able to take a step back and simply tell the rover what we're trying to find and have it report back when it finds something we'll think is interesting? Giving the rover this type of autonomy removes the need for constant interaction and makes the speed of light a moot point.

Hard-coding a procedure for handling all of the cases a rover could encounter while navigating — and the thousands of other tasks that a rover might have to undertake — is not a good option in these cases. The need for flexibility is key, and the on-board storage space is typically quite limited. Due to the large distances, communication lag, and changing mission parameters, any efforts in space explo-

 ISSN 0738-4602

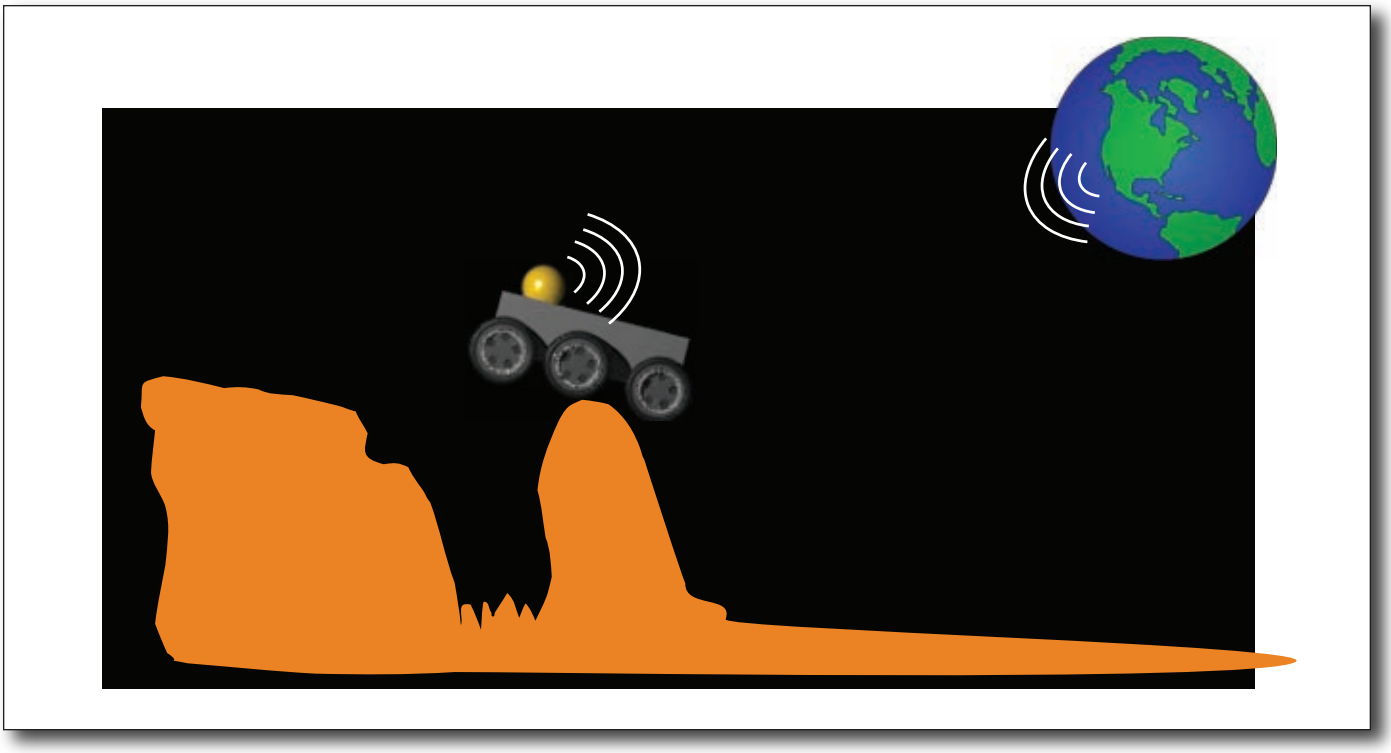


Figure 1. The Speed-of-Light Communication Delay Makes Artificial Intelligence a Necessity for Space Exploration.

ration need to be extremely robust to a wide array of possible disturbances and capable of a wide array of tasks. In short, as the human race expands its efforts to explore the solar system, artificial intelligence will play a key role in many high-level control decisions.

However, giving a rover that cost many person-years of labor and a multimillion dollar budget complete autonomy over its actions on another planet might be a bit unnerving. Space is a harsh and dangerous place; what if it isn't able to achieve the tasks it needs to? Worse, what if the rover finds an unpredicted and creative way to fail? These are legitimate concerns, worth addressing seriously.

One way to mitigate these concerns is to take the concept of a single traditional monolithic rover and broke it up into many pieces, creating a team of rovers, with one to embody each of these pieces. Each would be simple and perform just a few functions. Though each of the pieces is less effective individually than the monolithic rover, the sum of the pieces is greater than the whole in many ways.

First, any of the members of the team is significantly more expendable than the whole monolithic rover. This alleviates a large number of concerns and opens many opportunities. If one rover does find a way to fail creatively, the remainder of the team is still completely operational. By the same token, the team of rovers can undertake more dangerous missions than the monolithic rover; if the dangerous conditions lead to the failure of one rover, the rest can complete the mission. Additionally, redundancy can be designed into the team for particularly dangerous or critical roles.

Beyond the disposability of the individual team members, there are other benefits to this team-based approach. Savings can be realized in construction, as each rover can be designed with parts from a lower-cost parts portion of the reliability curve. Similar savings are available in the design process, as a new team can be formed with some members that have been previously designed.

In addition, a team of rovers can have capabilities that a single monolithic rover cannot, like having presence in multiple locations at once, which is incredibly useful for planetary exploration. Ephemeral events can be simultaneously observed from separate locations (Estlin et al. 2010), even from the ground and from orbit simultaneously (Chien et al. 2011), which can make interpreting the situation significantly easier. Construction tasks that might be impossible for a single rover with limited degrees of freedom become much easier. Teams can survey areas separated by impassible terrain and share long-range communication resources (Chien et al. 2000).

However, the concerns that we must address expand rapidly once we start to consider the possibilities that arise with multiple rovers acting in the same area simultaneously. How do the rovers coordinate so that their efforts lead to the maximum amount of interesting discoveries? How does a rover decide between achieving a task on its own versus helping another rover that has become stuck? How

does it decide between covering an area that's been deemed interesting or exploring an area that hasn't received much attention? These are all issues that fall under the larger umbrella of multiagent artificial intelligence (or multiagent systems), which is a ripe area of modern research (Wooldridge 2008).

One technique that has proven useful within the multiagent systems community is that of reward shaping used in conjunction with reinforcement learning. In this paradigm, instead of the rovers being told what to do, they each individually learn what to do through an iterative process of trial and error 1. In this process, each rover learns to maximize a reward function, measuring its performance. By carefully shaping the rewards that the rovers receive, we can promote coordination and improve the robustness of the learning process (Mataric 1994; Taylor and Stone 2009). Our goals in reward shaping are to balance two fundamental tensions in learning: (1) the rewards that the rovers are maximizing should be informative enough that they can promote coordination of the entire system, and (2) they should be simple enough that the rovers can easily determine the best actions to take to maximize their rewards. There are a number of obstacles that can make achieving this goal more difficult.

Multiagent Coordination Is Hard

Being able to automatically learn intelligent control policies for autonomous systems is an exciting prospect for space exploration. Especially within the context of a coordinated set of autonomous systems, we have the possibility of achieving increased capabilities while maintaining an adaptive and robust system. However, these multiagent systems are fundamentally different from other types of artificial intelligence in two ways. First, we have to promote coordination in a multiagent system (see figure 2), since agents learning by themselves may work at cross-purposes, and second, we have to overcome increased learning complexity as the actions taken by other agents increase the difficulty that any particular agent has in determining the value of its actions with respect to a coordinated goal.

In space applications, this coordination will involve many issues like optimizing communication networks, maximizing scientific information returned from a set of sensors, and coordinating power usage through shared power resources. As a guiding example, consider a group of autonomous rover agents set to explore an area of an extraterrestrial body. Their goal is to observe a series of points of interest, and gain as much knowledge about these points as possible on a teamwide level. This means that ideally each agent within the multiagent system will cooperate toward the common good, but how to do this is not immediately obvious. For example, it may not be readily apparent in practice that a rover is actively observing a point that has been well studied at an earlier point in time. The rover's actions of observing that point may be a very good choice, except that the other agents acting in the environment had already gleaned the necessary information from the point, making the action redundant.

Complex communication protocols or teamwork frameworks may offer a solution to this problem, but it might not be a practical one for space travel. Communication availability is limited, and failures of existing rovers or introduction of new rovers that weren't originally planned into the team are a realistic expectation for space exploration (Stone et al. 2013). Because of the large travel times and distances, and unpredictable and harsh environments, flexibility in implementation is key, and the solution must be robust to all sorts of disturbances.

This flexibility can be developed through the use of adaptive agent policies, which change over time to fit the situation the rover encounters. This creates a learning multiagent system, which allows the team to effectively deal with changing environments or mission parameters. A key issue in a learning multiagent system is the choice of the reward function that the agents use.

How to Judge a Reward Function

A multiagent learning system depends on a way to measure the value of each agent's behavior. For instance, did a particular sensor reading give additional scientific value? Did a particular message sent efficiently use the communications channel? Did a particular rover movement put the rover in a good location and not interfere with the actions of another rover? This measurement is called a reward function, and changing what form the reward function takes is the science of reward shaping (Chalkiadakis and Boutilier 2003; Guestrin, Lagoudakis, and Parr 2002; Hu and Wellman 1998; Mataric 1998; Stone and Veloso 2000; Tumer, Agogino, and Wolpert 2002; Wolpert and Tumer 2001). An agent will seek to solely increase its reward function. Thus it should have two specific properties: sensitivity and alignment.

First, the reward function must be sensitive to the actions of the agent (Wolpert and Tumer 2001). An agent taking good actions should receive a high reward, and an agent taking poor actions should receive a lower reward. In an unpredictable, stochastic, or multiagent environment, there are other factors affecting the reward that the agent will receive. An ill-developed reward function will allow these random factors to insert a large amount of noise into the signal offered by the reward function, and as the signal-to-noise ratio decreases, so does the agent's performance.

Second, the reward function must be aligned with the overall mission that the agent team must achieve (Wolpert and Tumer 2001). That is, an agent that

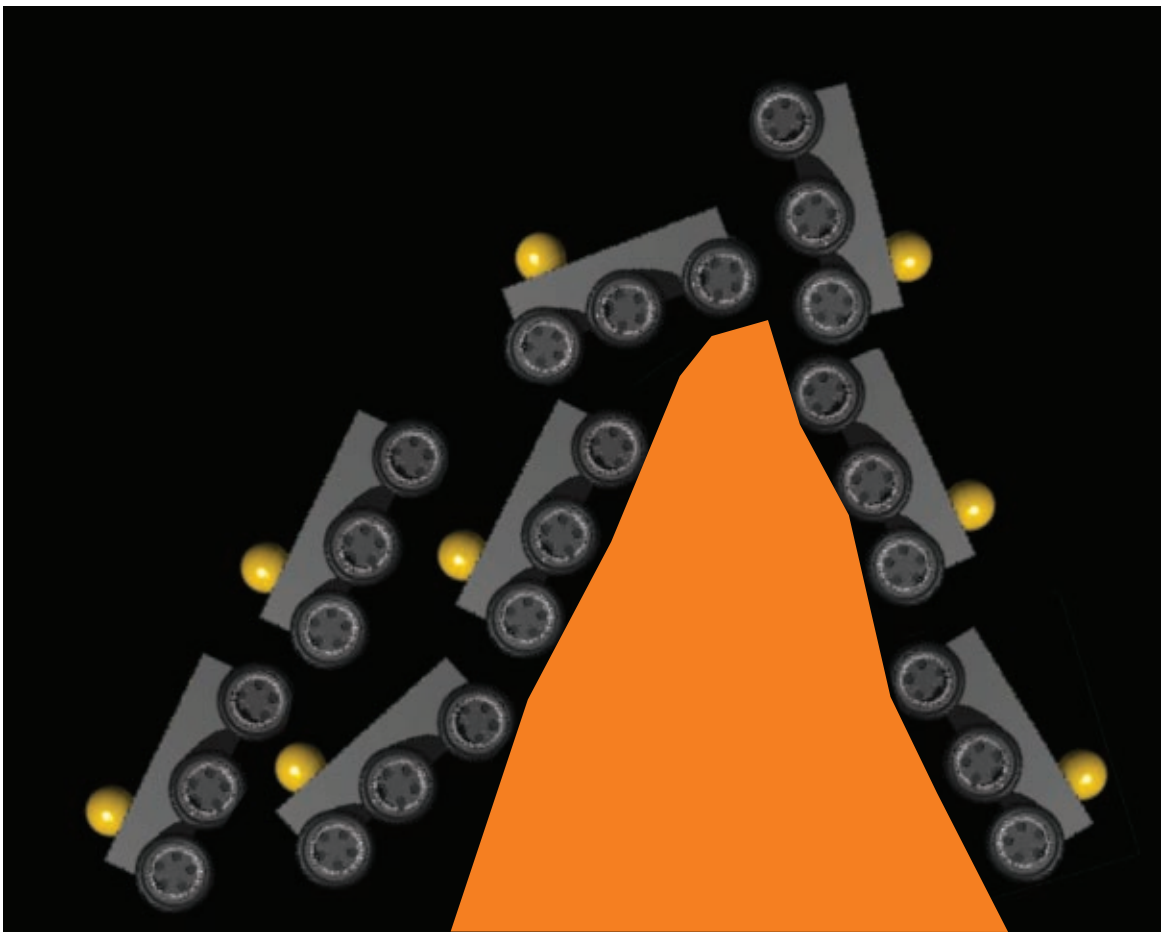


Figure 2. Effective Single-Agent Learning May Lead to Incompatible Interactions in a Multiagent Setting.

Repetitive exploration and congestion are common problems.

increases its own reward should simultaneously be increasing the system performance. A lack of alignment can lead to situations such as the tragedy of the commons (Hardin 1968, Crowe 1969), wherein a group of rationally self-concerned agents lead to a drop in system performance due to working at cross-purposes. That is, agent *A* does what it perceives in its own best interest, as does agent *B;* in some way, their actions deplete their shared environment and lead to both agents being worse off than they would be had they cooperated for the communal good.

Both of these properties — sensitivity and alignment — are critical to multiagent systems. An agent must be able to clearly discern what it has done to earn a high reward, and continuing to earn that high reward must be in the best interest of the system as a whole. This is especially the case in space applications, because the large distances and communication restrictions introduced by limited bandwidth, limited power, or line-of-sight lead-to time prevent outside intervention if the system performance were to go awry. In fact, even identifying that a problem exists within the system is challenging: space and extra planetary exploration is a complex and difficult problem, and it might not be easy to immediately diagnose when agents aren't achieving their full potential.

In this article, we show one approach to diagnosing potential system performance issues through visualizing the sensitivity and alignment of various reward structures in a simple and straightforward manner.

Classic Approaches to Coordination Through Reward Shaping

There are three classic approaches to solving complex multiagent systems: robot totalitarianism, robot socialism, and robot capitalism. Each has specific advantages and drawbacks.

Robot Totalitarianism (Centralized Control)

First, consider a centralized system in which one agent is making all necessary decisions for the entire system as a whole, and all other agents are merely following orders. The advantages here are that perfect coordination is possible and the pieces of the system as a whole will cooperate to increase system performance. This typically works well for small systems consisting of just a few agents (Sutton and Barto 1998). However, such a centralized system can fall prey to complexities such as communication restrictions, component failures — especially where a single point of failure can stop the entire system — and simply the difficulty of simultaneously solving a problem for hundreds or thousands of agents simultaneously. In most realistic situations, this is simply not an option.

Robot Socialism (Global or Team Reward)

Next, consider a system in which each agent is allowed to act autonomously in the way that it sees fit, and every agent is given the same global reward, which represents the system performance as a whole. They will single-mindedly pursue improvements on this reward, which means that their efforts are directed toward improving system performance, due to this reward having perfect alignment. However, because there may be hundreds or thousands of agents acting simultaneously in the shared environment, it may not be clear what led to the reward. In a completely linear system of n agents, each agent is only responsible for $1/n$ of the reward that they all receive, which can be entirely drowned out by the $(n - 1)/n$ portion for which that agent is not responsible. In a system with 100 agents, that means an agent might only have dominion over 1 percent of the reward it receives! This could lead to situations in which an agent chooses to do nothing, but the system reward increases, because other agents found good actions to take. This would encourage that agent to continue doing nothing, even though this hurts the system, due to a lack of sensitivity of the reward.

Robot Capitalism (Local or Perfectly Learnable Reward)

Finally, consider a system in which each agent has a local reward function related to how productive it is. For example, a planetary rover could be evaluated on how many photographs it captures of interesting rocks. This means that its reward is dependent only on itself, creating high sensitivity. However, the team of rovers obtaining hundreds of photographs of the same rock is not as interesting as obtaining hundreds of photographs of different rocks, though these would be evaluated the same with a local scheme. This means that the local reward is not aligned with the system-level reward.

Summary

Each of the reward functions has benefits and drawbacks that are closely mirrored in human systems. However, we are not limited to just these reward functions; as we mentioned before, an agent will single-mindedly seek to increase its reward, no matter what it is, whether or not this is in the best interest of the system at large. Is there, perhaps, a method that could be as aligned as the global reward, while as sensitive as the local reward, while still avoiding the pitfalls of the centralized approach?

Difference Rewards

An ideal solution would be to create a reward that is aligned with the system reward while removing the noise associated with other agents acting in the system. This would lead agents toward doing everything they can to improve the system's performance. Such a reward in a multirover system would reward a rover for taking a good action that coordinates well with rovers that are close to it, and would ignore the effects of distant rovers that were irrelevant.

A way to represent this analytically is to take the global reward $G(z)$ of the world z, and subtract off everything that doesn't have to do with the agent we're evaluating, revealing how much of a difference the agent made to the overall system. This takes the form

$$D_i(z) = G(z) - G(z_{-i}) \quad (1)$$

where $G(z_{-i})$ is the global reward of the world without the contributions of agent i, and $D_i(z)$ is the difference reward.

Let us first consider the alignment of this reward. $G(z)$ is perfectly aligned with the system reward. $G(z_{-i})$ may or may not be aligned, but in this case, it doesn't matter, because agent i (whom we are evaluating) has no impact on $G(z_{-i})$, by definition. This means that $D_i(z)$ is perfectly aligned, because all parts that agent i affects are aligned: agent i taking action to improve $D_i(z)$ will simultaneously improve $G(z)$.

Now, let us consider the sensitivity of this reward. $G(z)$ is as sensitive as the system reward, because it is identical. However, we remove $G(z_{-i})$ from the equation; that is, a large portion of the system — on which agent i has no impact on the performance — does not affect $D_i(z)$. This means that $D_i(z)$ is very sensitive to the actions of agent i and includes little noise from the actions of other agents.

Difference rewards are not a miracle cure. They do require additional computation to determine which

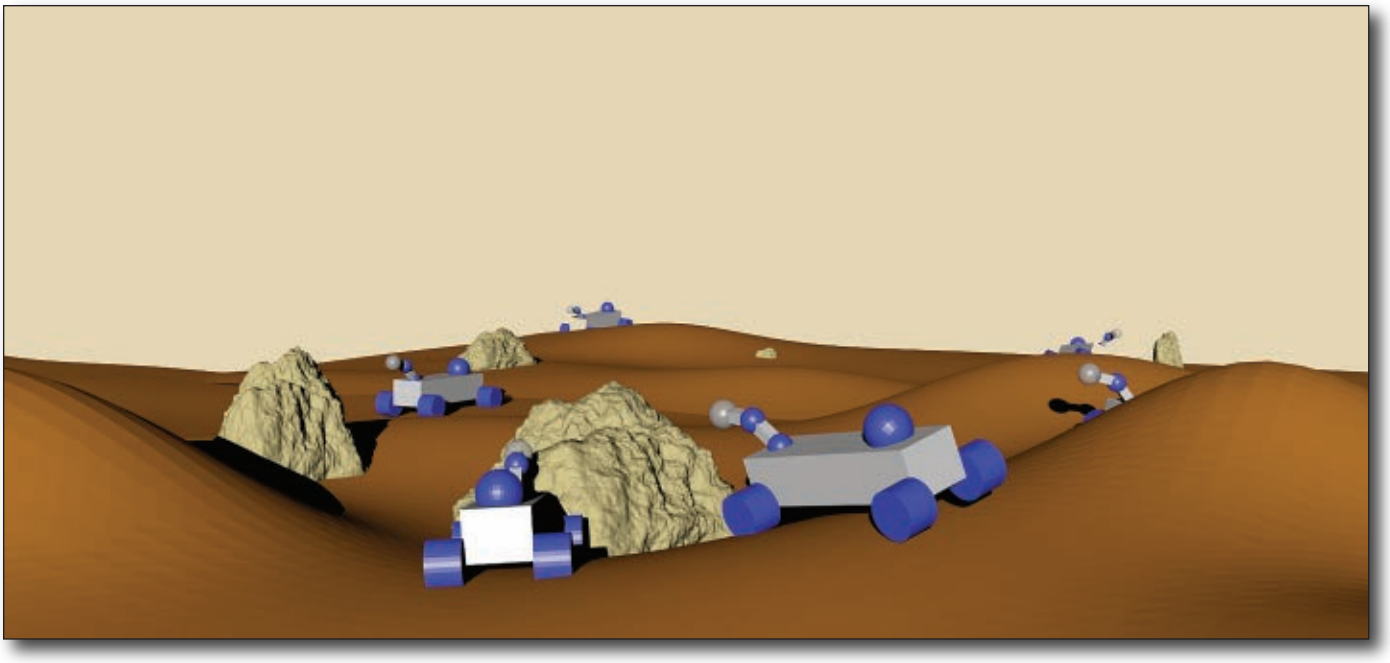


Figure 3. A Team of Rovers Exploring Various Points of Interest on the Martian Surface.

Artist's rendition.

portions of the system reward are caused by each agent. However, it is important to note that it is not necessary to analytically compute these contributions. In many cases, a simple approximation that serves to remove a large portion of the noise caused by using the system-level reward gains significant performance increases over using the system reward alone.

Although in this article we focus on the continuous rover domain, both the difference reward and the visualization approach have broad applicability. The difference reward used in this article has been applied to many domains, including data routing over a telecommunication network (Tumer and Wolpert 2000), multiagent gridworld (Tumer, Agogino, and Wolpert 2002), congestion games such as traffic toll lanes (Tumer and Wolpert 2004a, 2004b; Wolpert and Tumer 2001), and optimization problems such as bin packing (Wolpert, Tumer, and Bandari 2004) and faulty device selection (Tumer 2005).

Continuous Rover Domain

To examine the properties of the difference reward in a more practical way, let us return to our example of a team of rovers on a mission to explore an extraterrestrial body, like the moon or Mars (figure 3). We allow each rover to take continuous actions to move in the space, while receiving noisy sensor data at discrete time steps (Agogino and Tumer 2004).

Points of Interest

Certain points in the team's area of operation have been identified as points of interest (POIs), which we represent as green dots. Figure 4 offers one of the layouts of POIs that we studied, with a series of lower-valued POIs located to the left on the rectangular world, and a single high-valued POI located on the right half. Because multiple simultaneous observations of the same POI are not valued higher than a single observation in this domain, the best policy for the team is to spread out: one agent will closely study the large POI, while the remainder of the team will cover the smaller POIs on the other side.

Sensor Model

We assume that the rovers have the ability to sense the whole domain (except in the results we present later marked with PO for partial observability), but even so, using state variables to represent each of the rovers and POIs individually results in an intractable learning problem: there are simply too many parameters. This is also why a centralized controller does not function well in this case. We reduce the state space by providing eight inputs through the process illustrated in figure 5. For each quadrant, which rotates to remain aligned with the rover as it moves through the space, the rover has a rover sensor and a POI sensor. The rover sensor calculates the relative density and proximity of rovers within that quadrant and condenses this to a single value. The POI

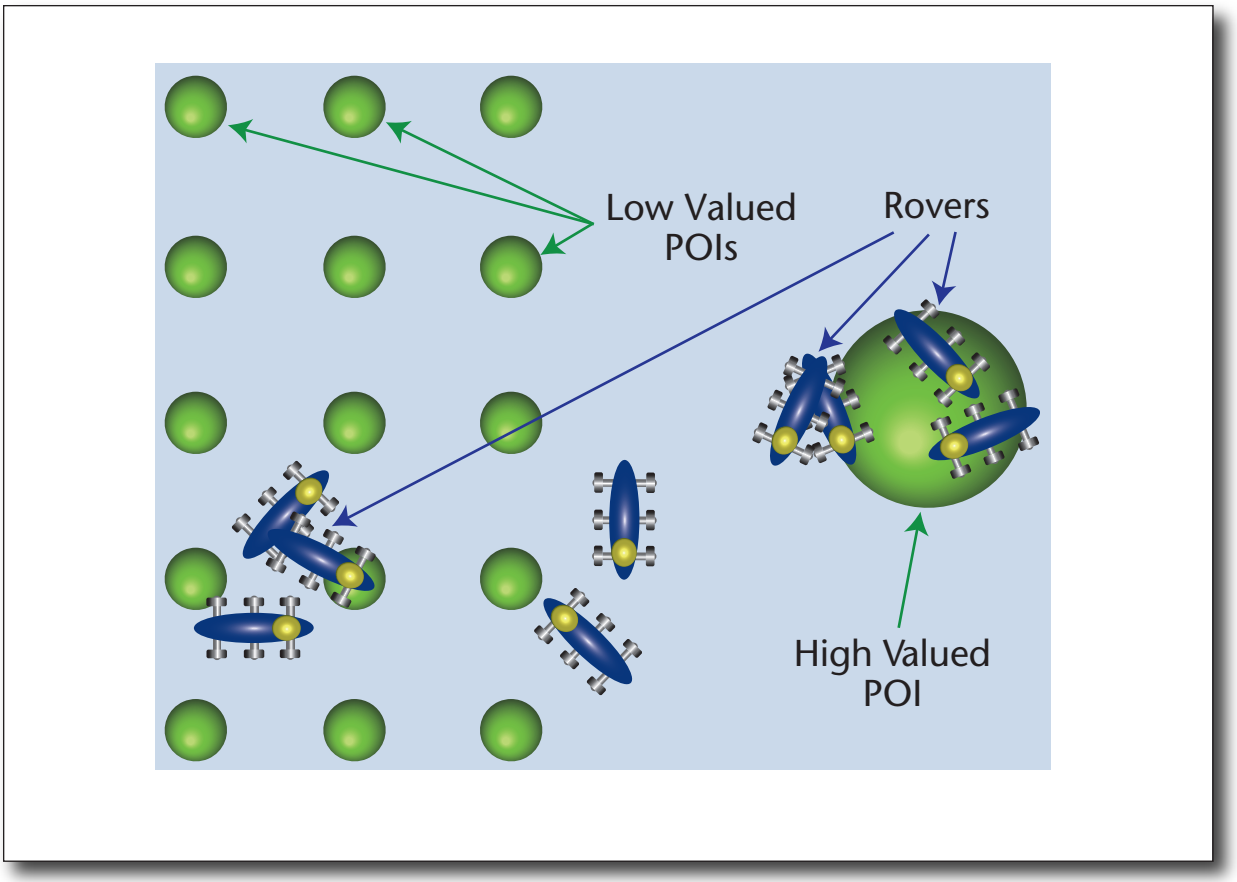


Figure 4. A Team of Rovers Observing a Set of Points of Interest.

Each POI has a value, represented by its size here. The team will ideally send one rover to observe the large POI on the right closely, while the rest spread out in the left region to observe as many small POIs as possible.

sensor does the same for all POIs within the quadrant.

Motion Model

We model the continuous motion of the rovers at each finite time step as shown in figure 6. We maintain the current heading of each rover, and at each time step the rovers select a value for *dy* and *dx*, where the value of *dy* represents how far forward the rover will move, and *dx* represents how much the rover will turn at that time step. The rover's heading for the next time step is represented as the direction of the resultant vector (*dx* + *dy*), shown as the solid line in figure 6.

Policy Search

The rovers use multilayer perceptrons (MLPs) with sigmoid activation functions to map the eight inputs provided by the four POI sensors and four rover sensors through 10 hidden units to two outputs, *dx* and *dy*, which govern the motion of the rover. The weights associated with the MLP are established through an online simulated annealing algorithm that changes the weights with preset probabilities (Kirkpatrick, Gelatt, and Vecchi 1983). This is a form of direct policy search, where the MLPs are the policies.

Reward Structures

We present the visualizations for alignment and sensitivity of four reward structures in this work. The *perfectly learnable local reward,* P_i, is calculated by considering the value of observations of all POIs made by agent *i* throughout the course of the simulation, ignoring the contributions that any other agents had to the system.

The global team reward, T_i, is calculated by considering the best observation the team as a whole made during the course of the simulation.

The *difference reward,* D_i, is calculated similarly to the perfectly learnable reward P_i, with the exception that if a second agent *j* also observed the POI, agent *i* is only rewarded with the difference between the quality of observations. Thus, if two agents observe a POI equally well, it adds to neither of their rewards, because the team would have observed it

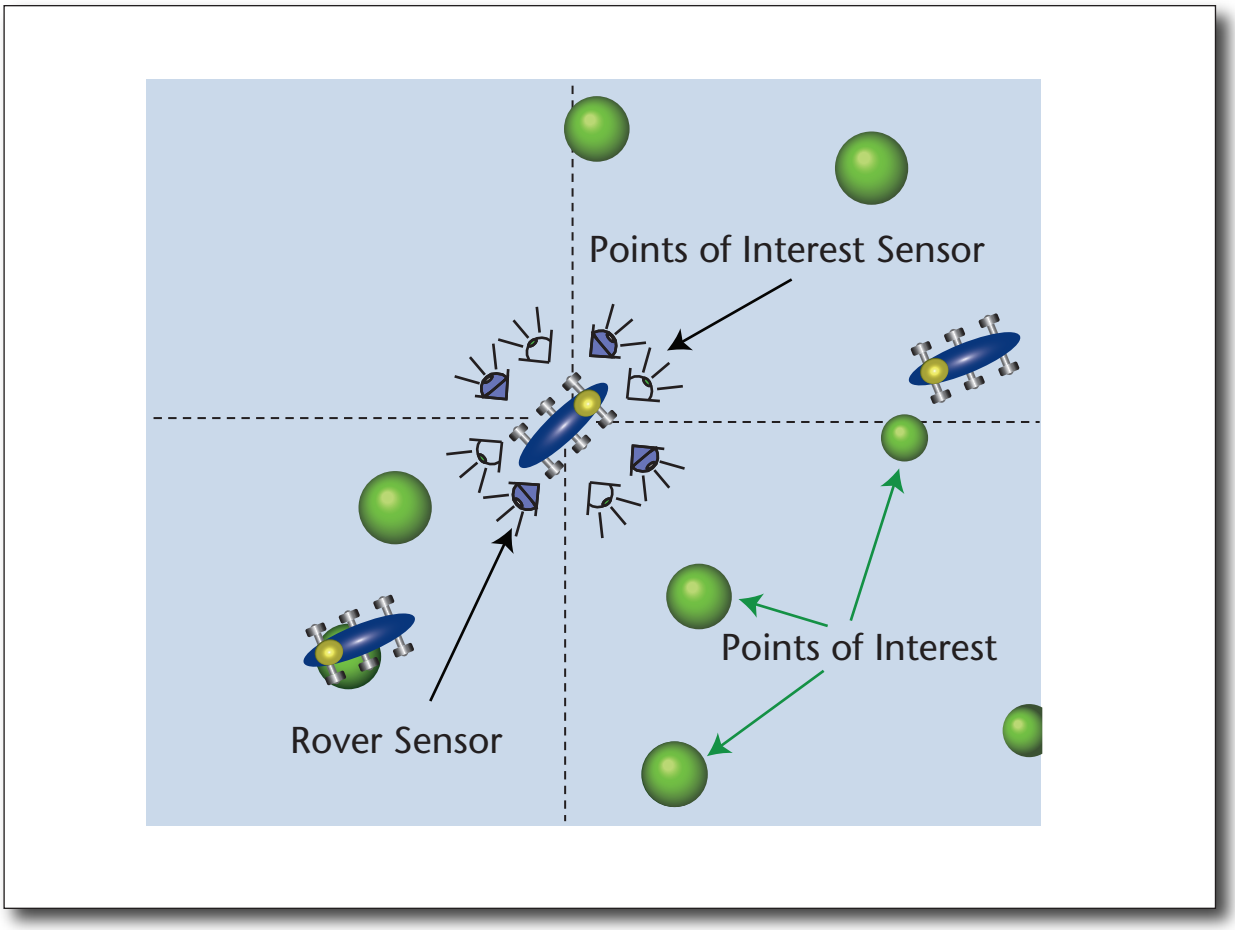


Figure 5. Rover Sensing Diagram.

Each rover has eight sensors: four rover sensors and four POI sensors that detect the relative congestion of each in each of the four quadrants that rotate with the rover as it moves.

anyway. If an agent is the sole observer of a POI, it gains the full value of the POI observation.

The difference reward under partial observability, D_i(PO), is calculated in the same manner as D_i, but with restrictions on what agent i can observe. Each rover evaluates itself in the same way as D_i, but because of the partial observability, it is possible that two rovers will be observing the same POI from opposite sides, and neither will realize that the POI is doubly observed (which does not increase the system performance), and both will credit themselves. Likewise, each rover cannot sense POIs located outside of its observation radius. This is represented in figure 7.

Visualization of Reward Structures

Visualization is an important part of understanding the inner workings of many systems, but particularly those of learning systems (Agogino, Martin, and Ghosh 1999; Bishof, Pinz, and Kropatsch 1992; Gallagher and Downs 1997; Hinton 1989; Hoen et al. 2004; Wejchert and Tesauro 1991). Especially in costly space systems we need additional validation that our learning systems are likely to work. Performance simulations can give us good performance bounds in scenarios that we can anticipate ahead of time. However, these simulations may not uniformly test the rovers in all situations that they may encounter. Learning and adaptation can allow rovers to adapt to unanticipated scenarios, but their reward functions still have to have high sensitivity and alignment to work. The visualization presented here can give us greater insight into the behavior of our reward functions. Our visualizations can answer important questions such as how often we think our reward will be aligned with our overall goals and how sensitive our rewards are to a rover's actions.

Through visual inspection we can see if there are important gaps in our coverage, and we can increase our confidence that a given reward system will work reliably.

The majority of the results presented in this work show the relative sensitivity and alignment of each of the reward structures. We have developed a unique method for visualizing these, which is illustrated in

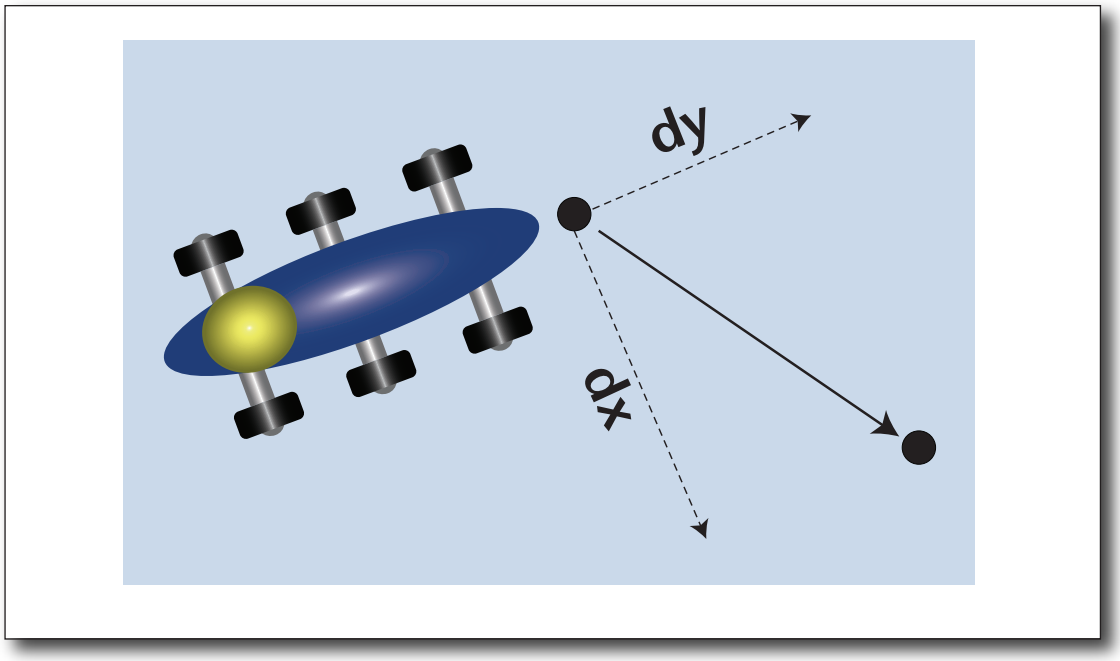


Figure 6. Rover Motion Model.

At each time step, each rover determines a continuous *dy* value to represent how far it moves in the direction it is facing, and a *dx* value determining how far it turns. Its heading at the next time step is the same as the vector (*dx* + *dy*).

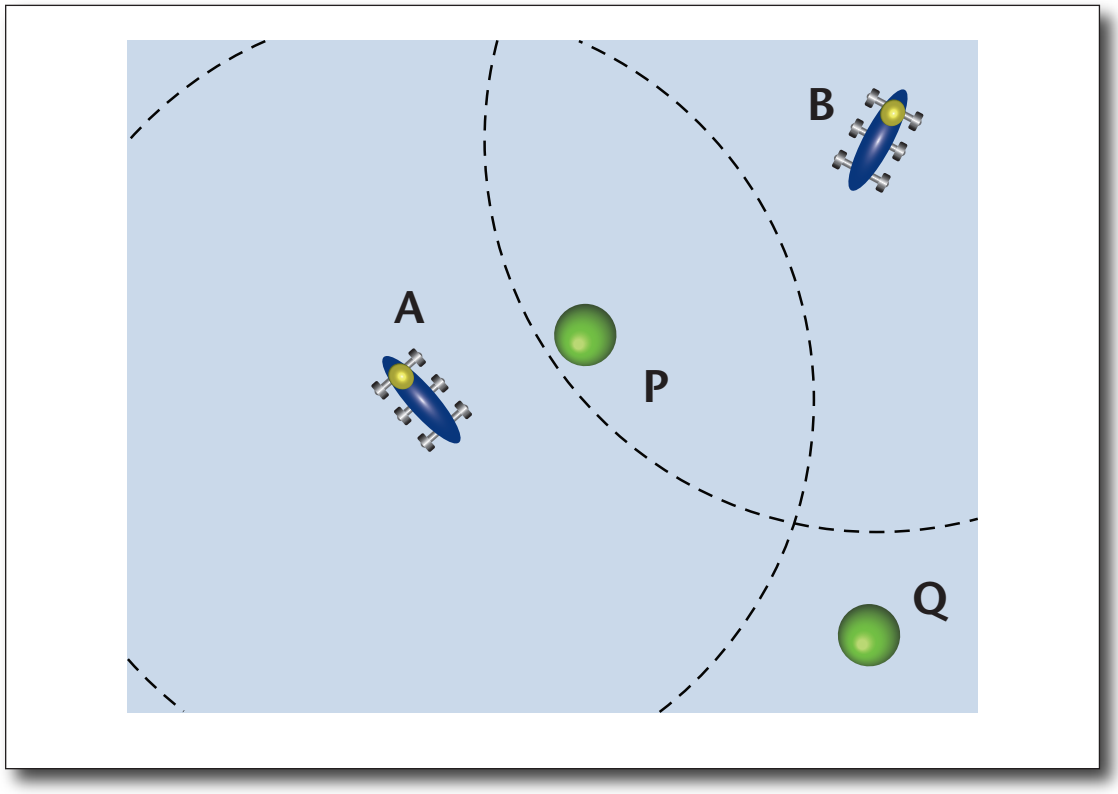


Figure 7. Rovers Under Partial Observability of Range Denoted by the Dotted Line.

Both rover A and rover B can sense and observe POI P, but cannot sense each other. In the Di(PO) formulation, they both would calculate that theirs was the only observation. Additionally, neither rover has any knowledge of POI Q.

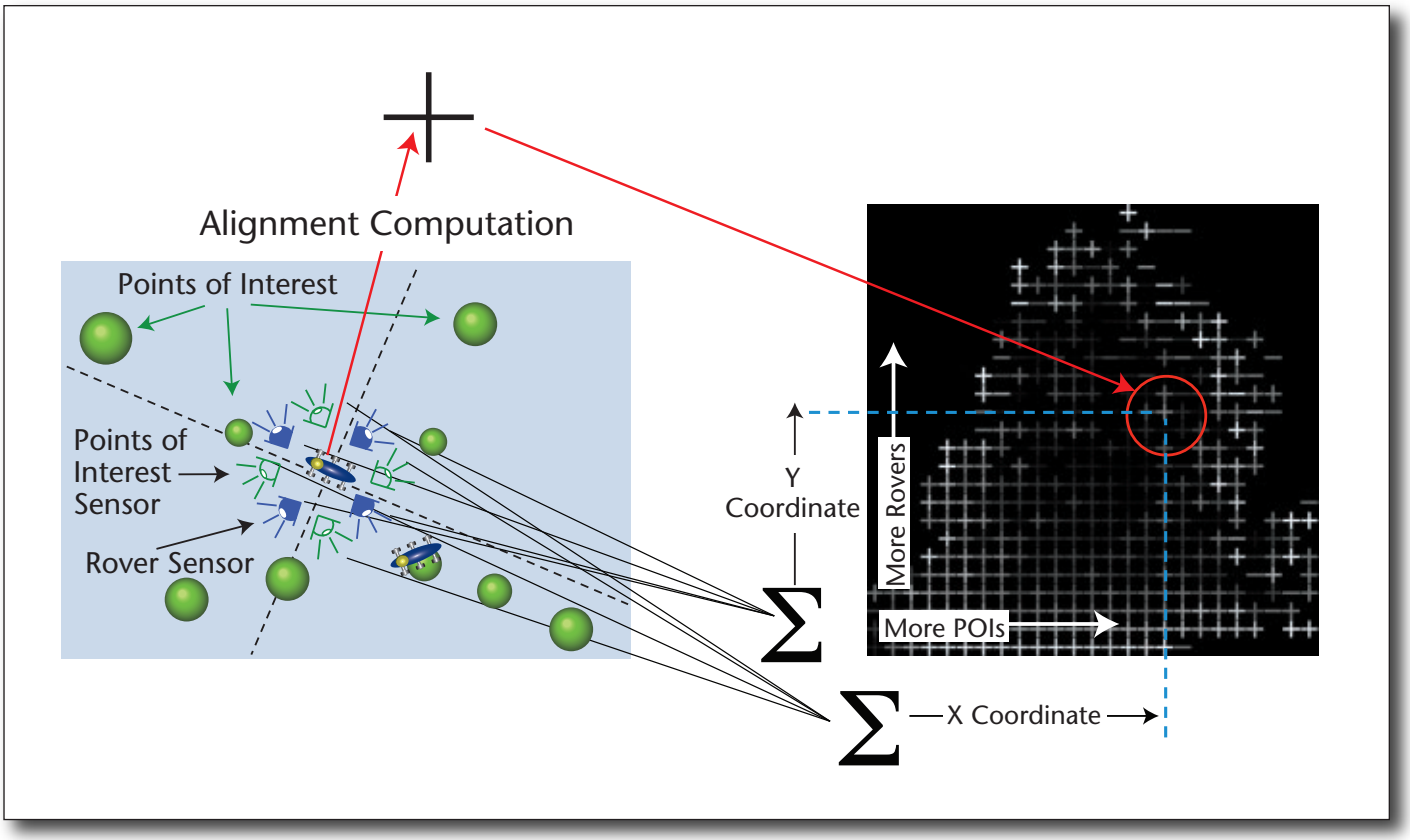


Figure 8. Illustration of the Visualization Calculation Process.

We use sensor data to determine which spot in the state space a circumstance represents, and place a marker in that location that represents whether the reward scores highly (bright +), near random (blank) or lowly (bright –).

Figure 8. We use the sensor information from the rover (left) to determine which of the spaces we will update (right). The alignment or sensitivity calculation (Agogino and Tumer 2008) is then represented by a symbol that takes the form of a "+" or "–" sign; the brighter the shade of the spot, the further from the average. A bright "+," then, represents a very aligned or very sensitive reward and a bright "–" represents an antialigned or very nonsensitive reward for a given POI and rover density, in the case of figure 9. We also present these calculations projected onto a specific case of the actual space that the rovers move through in figure 10. A more general version of this technique projects onto the principal components of the state space, which is more thoroughly explored in other work (Agogino and Tumer 2008).

Sensitivity and Alignment Analysis

A reward with simultaneously high alignment and sensitivity will be the easiest for agents to use to establish high-performing policies. Figure 9 presents the visualization for each of the reward structures. Notice that the perfectly learnable reward P_i does indeed have high sensitivity across the space, but has low alignment with the global reward in most of the center areas, which correspond to a moderate concentration of rovers and POIs. This area near the center of the visualization represents circumstances that the rovers find themselves in most often (Agogino and Tumer 2008).

The team reward T_i, by contrast, is very aligned throughout the search space, but is extremely lacking in sensitivity (denoted by the many "–" signs throughout the space).

The difference reward D_i is both highly aligned and highly sensitive throughout the search space. When we reduce the radius at which D_i can sense other rovers and POIs, the visualization from the $D_i(PO)$ row indicates that the sensitivity remains strong everywhere, but there is a slight drop in alignment throughout the space.

So, it would appear that difference rewards (D_i) offer benefits over other rewards, even with partial observability ($D_i(PO)$), but what does this mean in a more practical sense? To address this, we created figure 10, which projects the same type of alignment into the actual plane in which the rovers are operating.

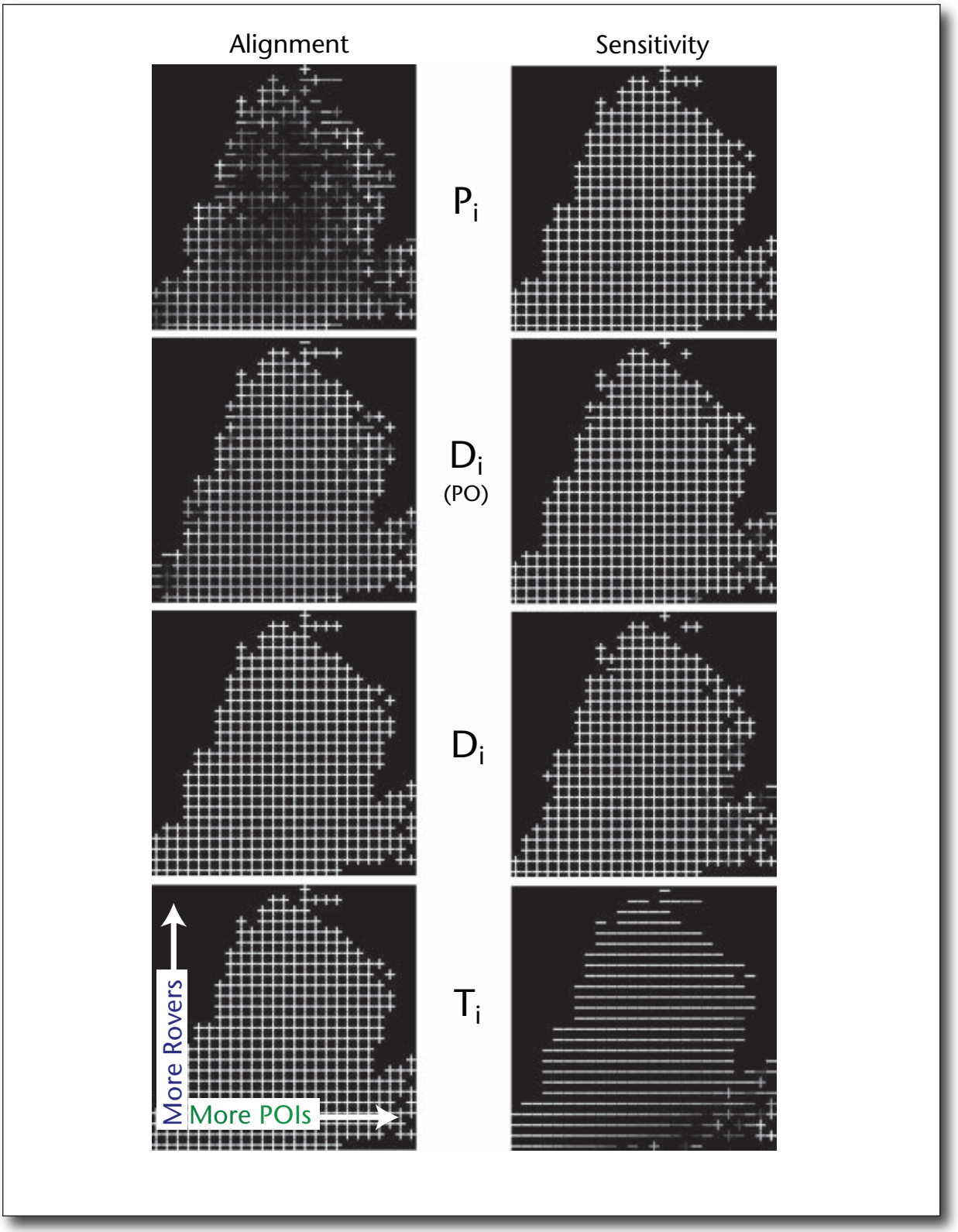


Figure 9. Alignment and Sensitivity Visualization for the Four Reward Types, Projected Onto a Two-Dimensional Space Representative of the State Space.

Note that the perfectly learnable reward Pi has low alignment through most of the space, and the team reward Ti is extremely nonsensitive through most of the space, while both instances of the difference reward maintain high performance by both metrics.

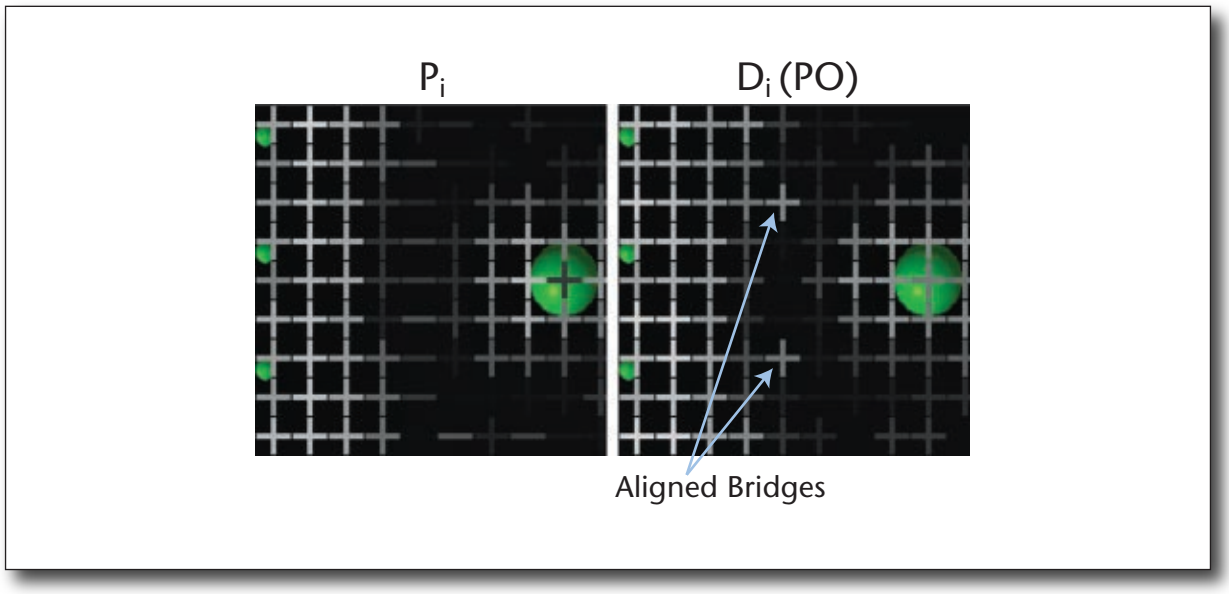


Figure 10. Alignment Visualization for the Perfectly Learnable Reward P_i, and the Difference Reward Under Partial Observability, Di(PO).

Projected onto the actual plane the rovers operate within.

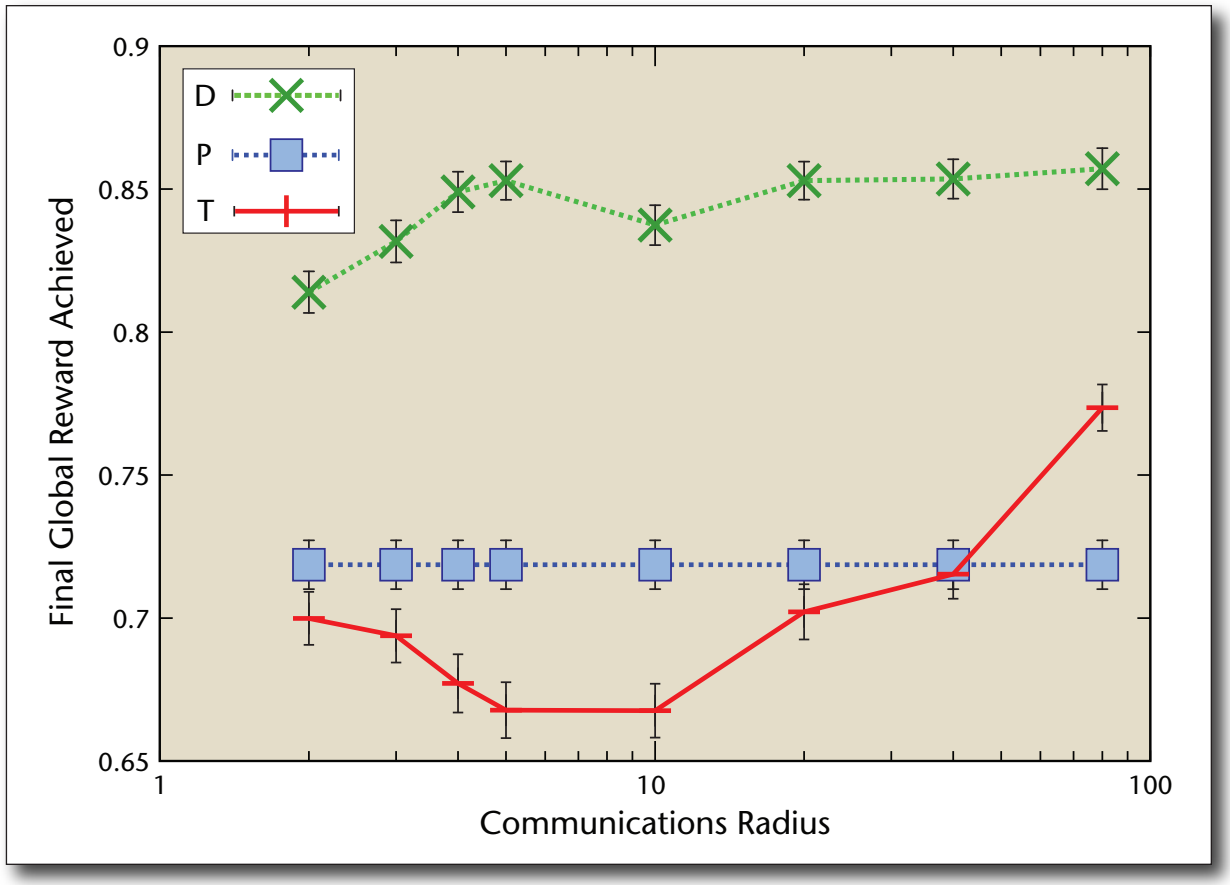


Figure 11. Final Performance Attained Versus Communication Radius for the Different Reward Structures.

Difference rewards maintain robust performance, but team rewards lose significant performance under restricted communication.

The left figure presents the alignment for the perfectly learnable reward P_i, and the indicated region is antialigned with the system-level reward. That is, even though traveling across this region would be beneficial to the team (because traveling across this region is required to reach the large POI on the right), the rovers that find themselves in this area of the space are actively penalized.

The figure on the right presents the alignment for the difference reward under observation restrictions $D_i(PO)$, which is qualitatively different within the highlighted regions: $D_i(PO)$ builds two aligned bridges, which allow the rovers to pass through the highlighted region without being penalized while they travel to the large POI on the right. Furthermore, the other parts of the highlighted region are not antialigned with the system reward meaning that the rovers are not penalized for traveling through this space; they merely do not increase their reward while there.

System Performance

We present system-level performance in figure 11, which represents the final system reward after training (y-axis) for teams of rovers trained on various rewards (line type), within different experiments with varying restrictions on observation radius (x-axis). Points to the left represent performance under extreme observation restrictions, and points to the right represent near-full observability. The visualizations performed in figures 9–10 correspond to full observability for all rewards except $D_i(PO)$, which corresponds to the D_i reward at a communication radius of 10 units in figure 11.

The benefits in sensitivity and alignment offered by the difference rewards D_i does result in increased system performance, as shown by the rightmost portion of figure 11. This reward leads to high-performing systems of rover agents with very successful policies. The global shared team reward T_i is capable of making some increases over a local policy under full observability, but still falls short of the difference reward.

The remainder of figure 11 presents a result based on the final system performance attained by agent teams operating with different rewards under restricted communications. Agents trained on the difference reward D_i are robust to a reduced communication radius, which could easily happen in cases of a dust storm, craggy landscape, or partial sensor failures. Agents using the perfectly learnable reward P_i are not affected by these restrictions, as the actions of other agents don't affect their policies.

Agents trained on the team or global reward T_i show an interesting phenomenon, however. Agents operating with a large communication radius are able to perform well as a team, and as this communication radius is reduced, so is the quality of the discovered policies — this much is expected. However, as the observation radius is decreased further, experimental runs with very low observation radii actually perform slightly better than those with moderate observation powers. This suggests that a little bit of knowledge about the location of other rovers is actually a bad thing. This can be explained: as the observation radius is reduced, agents trained on the team reward will behave more selfishly, like rovers using P_i, simply because they cannot sense the other rovers in the area; thus the gap between their performance decreases as the restrictions mirror this case.

Conclusions

Space exploration creates a unique set of challenges that must be addressed as we continue expanding our reach in the solar system. One approach for dealing with these challenges is through the use of reinforcement learning with reward shaping. Care must be taken in any use of reward shaping: a solution that works with a small number of agents will not necessarily scale up in an expected fashion and might lead to catastrophic system-level results. The readily obvious team reward and perfectly learnable reward both lead to poor results due to their low sensitivity and alignment, respectively. There is a need for local-level rewards that can be carried out quickly and efficiently that will scale into favorable results at the broader system level.

Difference rewards are an effective tool for this by encouraging multiagent coordination by their guaranteed alignment with the system objective, as well as their high sensitivity to local actions. They maintain high learnability throughout the state space, while offering perfect alignment with the system-level reward. This results in benefits that can be readily visualized within the space in which a team of rovers works, creating bridges of high reward that rovers can cross in between sparse POIs, and increasing overall system performance over a personal or team-based reward.

These properties in tandem with the robustness to various types of change within the environment show that their use in space exploration applications is an ideal fit. The capability of using a difference reward to encourage agents to do their best to help the team at whatever task is assigned allows for a team that can quickly and deftly adjust when mission parameters change. This can be as mundane as a sensor failing, or as dramatic as a complete mission reassignment.

While developing more sophisticated technologies for sensing more about the environment in a more efficient manner is a useful step forward, for multiagent space exploration, the key problem remains as what should the agents do to work together? This persists as a fertile motivating question for future research.

References

Agogino, A., and Tumer, K. 2004. Efficient Evaluation Functions for Multi-Rover Systems. In *Proceedings of the Genetic and Evolutionary Computation Conference* (GECCO-2004), Lecture Notes in Computer Science volume 3103, 1–12. Berlin: Springer.

Agogino, A.; Martin, C.; and Ghosh, J. 1999. Visualization of Radial Basis Function Networks. In *Proceedings of International Joint Conference on Neural Networks*. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Agogino, A. K., and Tumer, K. 2008. Analyzing and Visualizing Multiagent Rewards in Dynamic and Stochastic Environments. *Journal of Autonomous Agents and Multi-Agent Systems* 17(2): 320–338.

dx.doi.org/10.1007/s10458-008- 9046-9

Bishof, H.; Pinz, A.; and Kropatsch, W. G. 1992. Visualization Methods for Neural Networks. In *Proceedings of the 11th International Conference on Pattern Recognition,* 581–585. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Chalkiadakis, G., and Boutilier, C. 2003. Coordination in Multiagent Reinforcement Learning: A Bayesian Approach. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems* (AAMAS-03). New York: Association for Computing Machinery.

Chien, S.; Barrett, A.; Estlin, T.; and Rabideau, G. 2000. A Comparison of Coordinated Planning Methods for Cooperating Rovers. In *Proceedings of the Fourth International Conference on Autonomous Agents* (Agents '00). New York: Association for Computing Machinery.

Chien, S.; Doubleday, J.; Mclaren, D.; Tran, D.; Tanpipat, V.; Chitradon, R.; Boonyaaroonnef, S.; Thanapakpawin, P.; Khunboa, C.; Leelapatra, W.; Plermkamon, V.; Raghavendra, C.; Mandl, D. 2011. Combining Space-Based and In-Situ Measurements to Track Flooding in Thailand. In *Proceedings of the 2011 IEEE International Symposium on Geoscience and Remote Sensing,* 3935–3938. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Crowe, B. L. 1969. The Tragedy of the Commons Revisited. *Science* 166(3909)(November 28): 1103–1107.

Estlin, T. Chien, S.; Castano, R.; Doubleday, J.; Gaines, D.; Anderson, R. C.; de Granville, C.; Knight, R.; Rabideau, G.; Tang, B. 2010. Coordinating Multiple Spacecraft in Joint Science Campaigns. Paper Presented at the 10th International Symposium on Space Artificial Intelligence, Robotics, and Automation for Space (i-SAIRAS 2010). Sapporo, Japan. August 29–September 1.

Gallagher, M., and Downs, T. 1997. Visualization of Learning in Neural Networks Using Principal Component Analysis. Paper presented at the International Conference on Computational Intelligence and Multimedia Applications, Griffith University, Gold Coast, Australia, 10–12 February.

Guestrin, C.; Lagoudakis, M.; and Parr, R. 2002. Coordinated Reinforcement Learning. In *Proceedings of the 19th International Conference on Machine Learning*. San Francisco: Morgan Kaufmann Publishers.

Hardin, G. 1968. The Tragedy of the Commons. *Science* 162(3859)(December 13): 1243–1248.

Hinton, G. 1989. Connectionist Learning Procedures. *Artificial Intelligence* 40(1–3): 185–234. dx.doi.org/10.1016/0004-3702(89)90049

Hoen, P.; Redekar, G.; Robu, V.; and La Poutre, H. 2004. Simulation and Visualization of a Market-Based Model for Logistics Management in Transportation. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems,* 1218–1219. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Hu, J., and Wellman, M. P. 1998. Multiagent Reinforcement Learning: Theoretical Framework and an Algorithm. In *Proceedings of the Fifteenth International Conference on Machine Learning,* 242–250. San Francisco: Morgan Kaufmann, Inc.

Kirkpatrick, S.; Gelatt, C. D. J.; and Vecchi, M. P. 1983. Optimization by Simulated Annealing. *Science* 220(4598)(May 13): 671–680.

Mataric, M. J. 1998. New Directions: Robotics: Coordination and Learning in Multi-Robot Systems. *IEEE Intelligent Systems* 13(2): 6–8. dx.doi.org/10.1109/5254.671083

Mataric, M. J. 1994. Reward Functions for Accelerated Learning. In *Proceedings of the Eleventh International Conference on Machine Learning,* 181–189. San Francisco: Morgan Kaufmann, Inc.

Stone, P., and Veloso, M. 2000. Multiagent Systems: A Survey from a Machine Learning Perspective. *Autonomous Robots* 8(3): 345–383. dx.doi.org/10.1023/A:1008942012299

Stone, P.; Kaminka, G. A.; Kraus, S.; Rosenschein, J. R.; and Agmon, N. 2013. Teaching and Leading an Ad Hoc Teammate: Collaboration Without Pre-Coordination. *Artificial Intelligence* 203 (October): 35–65. dx.doi.org/10.1016/j.artint.2013.07.003

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. Cambridge, MA: The MIT Press.

Taylor, M. E., and Stone, P. 2009. Transfer Learning for Reinforcement Learning Domains: A Survey. *Journal of Machine Learning Research* 10 (2009): 1633–1685

Tumer, K. 2005. Designing Agent Utilities for Coordinated, Scalable, and Robust Multi-Agent Systems. In *Challenges in the Coordination of Large Scale Multiagent Systems,* ed. P. Scerri, R. Mailler, and R. Vincent. Berlin: Springer.

Tumer, K., and Wolpert, D. H. 2000. Collective Intelligence and Braess' Paradox. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence,* 104–109. Menlo Park, CA: AAAI Press.

Tumer, K., and Wolpert, D., eds. 2004a. *Collectives and the Design of Complex Systems*. Berlin: Springer. dx.doi.org/10.1007/978-1-4419-8909-3

Tumer, K., and Wolpert, D. 2004b. A Survey of Collectives. In *Collectives and the Design of Complex Systems,* ed. K. Tumer and D. Wolpert, 1–42. Berlin: Springer. dx.doi.org/10.1007/978-1-4419-8909-3_1

Tumer, K.; Agogino, A.; and Wolpert, D. 2002. Learning Sequences of Actions in Collectives of Autonomous Agents. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems,* 378–385. New York: Association for Computing Machinery. dx.doi.org/10.1145/544741.544832

Wejchert, J., and Tesauro, G. 1991. Visualizing Processes in Neural Networks. *IBM Journal of Research and Development* 35(1–2): 244–253. dx.doi.org/10.1147/rd.351.0244

Wolpert, D. H., and Tumer, K. 2001. Optimal Payoff Functions for Members of Collectives. *Advances in Complex Systems* 4(2/3): 265–279. dx.doi.org/10.1142/S0219525901000188

Wolpert, D. H.; Tumer, K.; and Bandari, E. 2004. Improving Search Algorithms by Using Intelligent Coordinates. *Physical Review E* 69:017701. dx.doi.org/10.1103/PhysRevE.69.017701

Wooldridge, M. 2008. *An Introduction to Multiagent Systems*. New York: Wiley.

Logan Yliniemi is a graduate research assistant at Oregon State University. He is pursuing his Ph.D. in robotics. His research focuses on credit assignment in multiagent systems and the interactions between multiagent systems and multiobjective problems.

Adrian Agogino is a researcher with the Robust Software Engineering Group in the Intelligent Systems Division at NASA Ames Research Center, employed through the University of California, Santa Cruz. His research interests are in the fields of machine learning, complex learning systems, and multiagent control.

Kagan Tumer is a professor of robotics and control at Oregon State University. His research interests are control and optimization in large autonomous systems with a particular emphasis on multiagent coordination. Applications of his work include coordinating multiple robots, optimizing large sensor networks, controlling unmanned aerial vehicles, reducing traffic congestion, and managing air traffic.

A Review of Real-Time Strategy Game AI

Glen Robertson, Ian Watson

■ *This literature review covers AI techniques used for real-time strategy video games, focusing specifically on StarCraft. It finds that the main areas of current academic research are in tactical and strategic decision making, plan recognition, and learning, and it outlines the research contributions in each of these areas. The paper then contrasts the use of game AI in academe and industry, finding the academic research heavily focused on creating game-winning agents, while the industry aims to maximize player enjoyment. It finds that industry adoption of academic research is low because it is either inapplicable or too time-consuming and risky to implement in a new game, which highlights an area for potential investigation: bridging the gap between academe and industry. Finally, the areas of spatial reasoning, multiscale AI, and cooperation are found to require future work, and standardized evaluation methods are proposed to produce comparable results between studies.*

Games are an ideal domain for exploring the capabilities of artificial intelligence (AI) within a constrained environment and a fixed set of rules, where problem-solving techniques can be developed and evaluated before being applied to more complex real-world problems (Schaeffer 2001). AI has notably been applied to board games, such as chess, Scrabble, and backgammon, creating competition that has sped the development of many heuristic-based search techniques (Schaeffer 2001). Over the past decade, there has been increasing interest in research based on video game AI, which was initiated by Laird and van Lent (2001) in their call for the use of video games as a test bed for AI research. They saw video games as a potential area for iterative advancement in increasingly sophisticated scenarios, eventually leading to the development of human-level AI. Buro (2003) later called for increased research in real-time strategy (RTS) games as they provide a sandbox for exploring various complex challenges that are central to game AI and many other problems.

Video games are an attractive alternative to robotics for AI research because they increasingly provide a complex and realistic environment for simulation, with few of the messy properties (and cost) of real-world equipment (Buro 2004; Laird and van Lent 2001). They also present a number of challenges that set them apart from the simpler board games that AI has famously been applied to in the past. Video games often have real-time constraints that prevent players from thinking extensively about each action, randomness that prevents players from completely planning future events, and hidden information that prevents players from

 ISSN 0738-4602

Figure 1. A Typical Match Start in an RTS Game.

Worker units have been sent to gather resources (right) and return them to the central building. Resources (recorded top right) are being spent building an additional worker (bottom center). Dark fog (left) blocks visibility away from player units.

knowing exactly what the other players are doing. Similar to many board games, competitive video games usually require adversarial reasoning to react according to other players' actions (Laird and van Lent 2001; Mehta et al. 2009; Weber, Mateas, and Jhala 2010).

RTS Games

This article is focused on real-time strategy games, which are essentially simplified military simulations. In an RTS game, a player indirectly controls many units and structures by issuing orders from an overhead perspective (figure 1) in real time in order to gather resources, build an infrastructure and an army, and destroy the opposing player's forces. The real-time aspect comes from the fact that players do not take turns, but instead may perform as many actions as they are physically able to make, while the game simulation runs at a constant frame rate (24 frames per second in StarCraft) to approximate a continuous flow of time. Some notable RTS games include *Dune II, Total Annihilation,* and the *Warcraft, Command & Conquer, Age of Empires,* and *StarCraft* series.

Generally, each match in an RTS game involves two players starting with a few units and/or structures in different locations on a two-dimensional terrain (map). Nearby resources can be gathered in order to produce additional units and structures and purchase upgrades, thus gaining access to more advanced in-game technology (units, structures, and upgrades). Additional resources and strategically important points are spread around the map, forcing players to spread out their units and buildings in order to attack or defend these positions. Visibility is usually limited to a small area around player-owned units, limiting information and forcing players to conduct reconnaissance in order to respond effectively to their opponents. In most RTS games, a

match ends when one player (or team) destroys all buildings belonging to the opponent player (or team), although often players will forfeit earlier when they see they cannot win.

RTS games have a variety of military units, used by the players to wage war, as well as units and structures to aid in resource collection, unit production, and upgrades. During a match, players must balance the development of their economy, infrastructure, and upgrades with the production of military units, so they have enough units to successfully attack and defend in the present and enough resources and upgrades to succeed later. They must also decide which units and structures to produce and which technologies to advance throughout the game in order to have access to the appropriate composition of units at the appropriate times. This long-term high-level planning and decision making, often called macromanagement, is referred to in this article as *strategic decision making*. In addition to strategic decision making, players must carefully control their units in order to maximize their effectiveness on the battlefield. Groups of units can be maneuvered into advantageous positions on the map to surround or escape the enemy, and individual units can be controlled to attack a weak enemy unit or avoid an incoming attack. This short-term control and decision making with individual units, often called micromanagement, and medium-term planning with groups of units, often called tactics, is referred to collectively in this article as *tactical decision making*.

In addition to the general video game challenges mentioned above, RTS games involve long-term goals and usually require multiple levels of abstraction and reasoning. They have a vast space of actions and game states, with durative actions, a huge branching factor, and actions that can have long-term effects throughout the course of a match (Buro and Churchill 2012; Buro and Furtak 2004; Mehta et al. 2009; Ontañón 2012; Tozour 2002; Weber, Mateas, and Jhala 2010). Even compared with Go, which is currently an active area of AI research, RTS games present a huge increase in complexity — at least an order of magnitude increase in the number of possible game states, actions to choose from, actions per game, and actions per minute (using standard rules) (Buro 2004; Schaeffer 2001; Synnaeve and Bessière 2011b). The state space is so large that traditional heuristic-based search techniques, which have proven effective in a range of board games (Schaeffer 2001), have so far been unable to solve all but the most restricted subproblems of RTS AI. Due to their complexity and challenges, RTS games are probably the best current environment in which to pursue Laird and van Lent's vision of game AI as a stepping stone toward human-level AI. It is a particularly interesting area for AI research because even the best agents are outmatched by experienced humans (Huang 2011; Synnaeve and Bessière 2011a; Weber, Mateas, and Jhala 2010), due to the human abilities to abstract, reason, learn, plan, and recognize plans (Buro 2004; Buro and Churchill 2012).

StarCraft

This article primarily examines AI research within a subtopic of RTS games: the RTS game StarCraft[1] (figure 2). StarCraft is a canonical RTS game, like chess is to board games, with a huge player base and numerous professional competitions. The game has three different but very well balanced teams, or races, allowing for varied strategies and tactics without any dominant strategy, and requires both strategic and tactical decision making roughly equally (Synnaeve and Bessière 2011b). These features give StarCraft an advantage over other RTS titles that are used for AI research, such as Wargus[2] and ORTS.[3]

StarCraft was chosen because of its increasing popularity for use in RTS game AI research, driven by the Brood War application programming interface (BWAPI)[4] and the AIIDE[5] and CIG[6] StarCraft AI Competitions. BWAPI provides an interface to programmatically interact with StarCraft, allowing external code to query the game state and execute actions as if they were a player in a match. The competitions pit StarCraft AI agents (or bots) against each other in full games of StarCraft to determine the best bots and improvements each year (Buro and Churchill 2012). Initially these competitions also involved simplified challenges based on subtasks in the game, such as controlling a given army to defeat an opponent with an equal army, but more recent competitions have used only complete matches. For more detail on StarCraft competitions and bots, see Ontañón et al. (in press).

In order to develop AI for StarCraft, researchers have tried many different techniques, as outlined in table 1. A community has formed around the game as a research platform, enabling people to build on each other's work and avoid repeating the necessary groundwork before an AI system can be implemented.

This work includes a terrain analysis module (Perkins 2010), well-documented source code for a complete, modular bot (Churchill and Buro 2012), and preprocessed data sets assembled from thousands of professional games (Synnaeve and Bessière 2012). StarCraft has a lasting popularity among professional and amateur players, including a large professional gaming scene in South Korea, with international competitions awarding millions of dollars in prizes every year (Churchill and Buro 2011). This popularity means that there are a large number of high-quality game logs (replays) available on the Internet that can be used for data mining, and there are many players of all skill levels to test against (Buro and Churchill 2012; Synnaeve and Bessière 2011b; Weber, Mateas, and Jhala 2011a).

This article presents a review of the literature on

Figure 2. Part of a Player's Base in StarCraft.

The white rectangle on the minimap (bottom left) is the area visible on screen. The minimap shows areas that are unexplored (black), explored but not visible (dark), and visible (light). It also shows the player's forces (lighter dots) and last-seen enemy buildings (darker dots).

RTS AI with an emphasis on StarCraft. It includes particular research based on other RTS games in the case that significant literature based on StarCraft is not (yet) available in that area. The article begins by outlining the different AI techniques used, grouped by the area in which they are primarily applied. These areas are tactical decision making, strategic decision making, plan recognition, and learning. This is followed by a comparison of the way game AI is used in academe and the game industry, which outlines the differences in goals and discusses the low adoption of academic research in the industry. Finally, some areas are identified in which there does not seem to be sufficient research on topics that are well-suited to study in the context of RTS game AI. This last section also calls for standardization of the evaluation methods used in StarCraft AI research in order to make comparison possible between papers.

Tactical Decision Making

Tactical and micromanagement decisions — controlling individual units or groups of units over a short period of time — often make use of a different technique from the AI that makes strategic decisions. These tactical decisions can follow a relatively simple metric, such as attempting to maximize the amount of enemy firepower that can be removed from the playing field in the shortest time (Davis 1999). In the video game industry, it is common for simple techniques, such as finite state machines, to be used to make these decisions (Buckland 2005). However, even in these small-scale decisions, many factors can

Figure 3. A Battle in StarCraft.

Intense micromanagement is required to maximize the effectiveness of individual units, especially spellcaster units like the Protoss Arbiter.

be considered to attempt to make the best decisions possible, particularly when using units with varied abilities (figure 3), but the problem space is not nearly as large as that of the full game, making feasible exploratory approaches to learning domain knowledge (Weber and Mateas 2009). There appears to be less research interest in this aspect of RTS game AI than in the area of large-scale, long-term strategic decision making and learning.

Reinforcement Learning

Reinforcement learning (RL) is an area of machine learning in which an agent must learn, by trial and error, optimal actions to take in particular situations order to maximize an overall reward value (Sutton and Barto 1998). Through many iterations of weakly supervised learning, RL can discover new solutions that are better than previously known solutions. It is relatively simple to apply to a new domain, as it requires only a description of the situation and possible actions, and a reward metric (Manslow 2004). However, in a domain as complex as an RTS game — even just for tactical decision making — RL often requires clever state abstraction mechanisms in order to learn effectively. This technique is not commonly used for large-scale strategic decision making, but is often applied to tactical decision making in RTS games, likely because of the huge problem space and delayed reward inherent in strategic decisions, which make RL difficult.

RL has been applied to StarCraft by Shantia, Begue, and Wiering (2011), where Sarsa, an algorithm for solving RL problems, is used to learn to control units in small skirmishes. They made use of

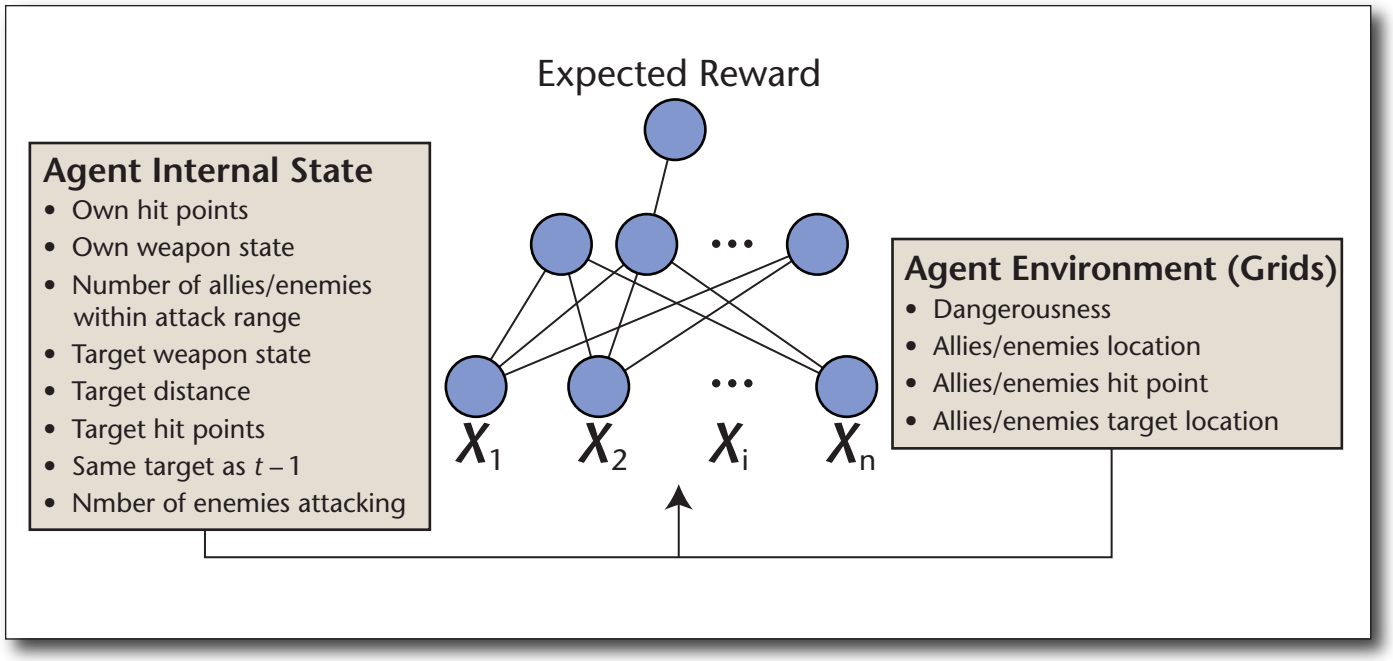


Figure 4. Game State Information Fed into a Neural Network to Produce an Expected Reward Value for a Particular Action.

Adapted from Shantia, Begue, and Wiering (2011).

Tactical Decision Making	Strategic Decision Making and Plan Recognition
Reinforcement Learning	Case-Based Planning
Game-Tree Search	Hierarchical Planning
Bayesian models	Behavior Trees
Case-Based Reasoning	Goal-Driven Autonomy
Neural Networks	State Space Planning
	Evolutionary Algorithms
	Cognitive Architectures
	Deductive Reasoning
	Probabilistic Reasoning
	Case-Based Reasoning

Table 1. AI Techniques Used for StarCraft.

artificial neural networks to learn the expected reward for attacking or fleeing with a particular unit in a given state (figure 4), and chose the action with the highest expected reward when in-game. The system learned to beat the inbuilt StarCraft AI scripting on average in only small three-unit skirmishes, with none of the variations learning to beat the in-built scripting on average in six-unit skirmishes (Shantia, Begue, and Wiering 2011).

RL techniques have also been applied to other RTS games. Sharma et al. (2007) and Molineaux, Aha, and Moore (2008) combine case-based reasoning (CBR) and RL for learning tactical-level unit control in MadRTS[7] (a description of CBR is presented later on in this article). Sharma et al. (2007) was able to increase the learning speed of the RL agent by beginning learning in a simple situation and then gradually increasing the complexity of the situation. The resulting performance of the agent was the same or better than an agent trained in the complex situation directly.

Their system stores its knowledge in cases that pertain to situations it has encountered before, as in CBR. However, each case stores the expected utility for every possible action in that situation as well as the contribution of that case to a reward value, allowing the system to learn desirable actions and situations. It remains to be seen how well it would work in a more complex domain.

Molineaux, Aha, and Moore (2008) describe a system for RL with nondiscrete actions. Their system retrieves similar cases from past experience and estimates the result of applying each case's actions to the current state. It then uses a separate case base to estimate the value of each estimated resulting state, and extrapolates around, or interpolates between, the actions to choose one that is estimated to provide the maximum value state. This technique results in a significant increase in performance when compared with one using discrete actions (Molineaux, Aha, and Moore 2008).

Human critique is added to RL by Judah et al. (2010) in order to learn tactical decision making for

controlling a small group of units in combat in Wargus. By interleaving sessions of autonomous state space exploration and human critique of the agent's actions, the system was able to learn a better policy in a fraction of the training iterations compared with using RL alone. However, slightly better overall results were achieved using human critique only to train the agent, possibly due to humans giving better feedback when they can see an immediate result (Judah et al. 2010).

Marthi et al. (2005) argues that it is preferable to decrease the apparent complexity of RTS games and potentially increase the effectiveness of RL or other techniques by decomposing the game into a hierarchy of interacting parts. Using this method, instead of coordinating a group of units by learning the correct combination of unit actions, each unit can be controlled individually with a higher-level group control affecting each individual's decision. Similar hierarchical decomposition appears in many RTS AI approaches because it reduces complexity from a combinatorial combination of possibilities — in this case, possible actions for each unit — down to a multiplicative combination.

Game-Tree Search

Search-based techniques have so far been unable to deal with the complexity of the long-term strategic aspects of RTS games, but they have been successfully applied to smaller-scale or abstracted versions of RTS combat. To apply these search methods, a simulator is usually required to allow the AI system to evaluate the results of actions very rapidly in order to explore the game tree.

Sailer, Buro, and Lanctot (2007) take a game theoretic approach by searching for the Nash equilibrium strategy among a set of known strategies in a simplified RTS. Their simplified RTS retains just the tactics aspect of RTS games by concentrating on unit group movements, so it does not require long-term planning for building infrastructure and also excludes micromanagement for controlling individual units. They use a simulation to compare the expected outcome from using each of the strategies against their opponent, for each of the strategies their opponent could be using (which is drawn from the same set), and select the Nash-optimal strategy. The simulation can avoid simulating every time step, skipping instead to just the states in which something interesting happens, such as a player making a decision, or units coming into firing range of opponents. Through this combination of abstraction, state skipping, and needing to examine only the possible moves prescribed by a pair of known strategies at a time, it is usually possible to search all the way to an end-game state very rapidly, which in turn means a simple evaluation function can be used. The resulting Nash player was able to defeat each of the scripted strategies, as long as the set included a viable counterstrategy for each strategy, and it also produced better results than the max-min and min-max players (Sailer, Buro, and Lanctot 2007).

Search-based techniques are particularly difficult to use in StarCraft because of the closed-source nature of the game and inability to arbitrarily manipulate the game state. This means that the precise mechanics of the game rules are unclear, and the game cannot be easily set up to run from a particular state to be used as a simulator. Furthermore, the game must carry out expensive calculations such as unit vision and collisions, and cannot be forced to skip ahead to just the interesting states, making it too slow for the purpose of search (Churchill, Saffidine, and Buro 2012). In order to overcome these problems, Churchill, Saffidine, and Buro (2012) created a simulator called SparCraft[8] that models StarCraft and approximates the rules, but allows the state to be arbitrarily manipulated and unnecessary expensive calculations to be ignored (including skipping uninteresting states). Using this simulator and a modified version of alpha-beta search, which takes into consideration actions of differing duration, they could find effective moves for a given configuration of units. Search time was limited to approximate real-time conditions, so the moves found were not optimal. This search allowed them to win an average of 92 percent of randomized balanced scenarios against all of the standard scripted strategies they tested against within their simulator (Churchill, Saffidine, and Buro 2012).

Despite working very well in simulation, the results do not translate perfectly back to the actual game of StarCraft, due to simplifications, such as the lack of unit collisions and acceleration, that affect the outcome (Churchill and Buro 2012; Churchill, Saffidine, and Buro 2012). The system was able to win only 84 percent of scenarios against the built in StarCraft AI despite the simulation predicting 100 percent, faring the worst in scenarios that were set up to require hit-and-run behavior (Churchill and Buro 2012). The main limitation of this system is that due to the combinatorial explosion of possible actions and states as the number of units increases, the number of possible actions in StarCraft, and a time constraint of 5ms per game frame, the search will only allow up to eight units per side in a two-player battle before it is too slow. On the other hand, better results may be achieved through opponent modeling, because the search can incorporate known opponent actions instead of searching through all possible opponent actions.

When this was tested on the scripted strategies with a perfect model of each opponent (the scripts themselves), the search was able to achieve at least a 95 percent win rate against each of the scripts in simulation (Churchill, Saffidine, and Buro 2012).

Monte Carlo Planning

Monte Carlo planning has received significant atten-

tion recently in the field of computer Go, but seems to be almost absent from RTS AI, and (to the authors' knowledge) completely untested in the domain of StarCraft. It involves sampling the decision space using randomly generated plans in order to find out which plans tend to lead to more successful outcomes. It may be very suitable for RTS games because it can deal with uncertainty, randomness, large decision spaces, and opponent actions through its sampling mechanism. Monte Carlo planning has likely not yet been applied to StarCraft due to the unavailability of an effective simulator, as was the case with the search methods above, as well as the complexity of the domain. However, it has been applied to some very restricted versions of RTS games. Although both of the examples seen here are considering tactical- and unit-level decisions, given a suitable abstraction and simulation, Monte Carlo tree search (MCTS) may also be effective at strategic level decision making in a domain as complex as StarCraft.

Chung, Buro, and Schaeffer (2005) created a capture-the-flag game in which each player needed to control a group of units to navigate through obstacles to the opposite side of a map and retrieve the opponent's flag. They created a generalized Monte Carlo planning framework and then applied it to their game, producing positive results. Unfortunately, they lacked a strong scripted opponent to test against, and their system was also very reliant on heuristic evaluations of intermediate states in order to make planning decisions. Later, Balla and Fern (2009) applied the more recent technique of upper confidence bounds applied to trees (UCT) to a simplified Wargus scenario. A major benefit of their approach is that it does not require a heuristic evaluation function for intermediate states, and instead plays a game randomly out to a terminal state in order to evaluate a plan. The system was evaluated by playing against a range of scripts and a human player in a scenario involving multiple friendly and enemy groups of the basic footman unit placed around an empty map. In these experiments, the UCT system made decisions at the tactical level for moving groups of units while micromanagement was controlled by the inbuilt Wargus AI, and the UCT evaluated terminal states based on either unit hit points remaining or time taken. The system was able to win all of the scenarios, unlike any of the scripts, and to overall outperform all of the other scripts and the human player on the particular metric (either hit points or time) that it was using.

Other Techniques

Various other AI techniques have been applied to tactical decision making in StarCraft. Synnaeve and Bessière (2011b) combine unit objectives, opportunities, and threats using a Bayesian model to decide which direction to move units in a battle. The model treats each of its sensory inputs as part of a probability equation that can be solved, given data (potentially learned through RL) about the distributions of the inputs with respect to the direction moved, to find the probability that a unit should move in each possible direction. The best direction can be selected, or the direction probabilities can be sampled over to avoid having two units choose to move into the same location. Their Bayesian model is paired with a hierarchical finite state machine to choose different sets of behavior for when units are engaging or avoiding enemy forces, or scouting. The bot produced was very effective against the built-in StarCraft AI as well as its own ablated versions (Synnaeve and Bessière 2011b).

CBR, although usually used for strategic reasoning in RTS AI, has also been applied to tactical decision making in Warcraft III,[9] a game that has a greater focus on micromanagement than StarCraft (Szczepanski and Aamodt 2009). CBR generally selects the most similar case for reuse, but Szczepanski and Aamodt (2009) added a conditional check to each case so that it could be selected only when its action was able to be executed. They also added reactionary cases that would be executed as soon as certain conditions were met. The resulting agent was able to beat the built in AI of Warcraft III in a micromanagement battle using only a small number of cases, and was able to assist human players by micromanaging battles to let the human focus on higher-level strategy.

Neuroevolution is a technique that uses an evolutionary algorithm to create or train an artificial neural network. Gabriel, Negru, and Zaharie (2012) use a neuroevolution approach called rtNEAT to evolve both the topology and connection weights of neural networks for individual unit control in StarCraft. In their approach, each unit has its own neural network that receives input from environmental sources (such as nearby units or obstacles) and hand-defined abstractions (such as the number, type, and quality of nearby units), and outputs whether to attack, retreat, or move left or right. During a game, the performance of the units is evaluated using a hand-crafted fitness function, and poorly performing unit agents are replaced by combinations of the best-performing agents. It is tested in very simple scenarios of 12 versus 12 units in a square arena, where all units on each side are either a hand-to-hand or ranged type unit. In these situations, it learns to beat the built-in StarCraft AI and some other bots. However, it remains unclear how well it would cope with more units or mixes of different unit types (Gabriel, Negru, and Zaharie 2012).

Strategic Decision Making

In order to create a system that can make intelligent actions at a strategic level in an RTS game, many researchers have created planning systems. These systems are capable of determining sequences of actions

to be taken in a particular situation in order to achieve specified goals. It is a challenging problem because of the incomplete information available — "fog of war" obscures areas of the battlefield that are out of sight of friendly units — as well as the huge state and action spaces and many simultaneous nonhierarchical goals. With planning systems, researchers hope to enable AI to play at a humanlike level, while simultaneously reducing the development effort required when compared with the scripting commonly used in industry. The main techniques used for planning systems are case-based planning (CBP), goal-driven autonomy (GDA) and hierarchical planning.

A basic strategic decision-making system was produced in-house for the commercial RTS game Kohan II: Kings of War[10] (Dill 2006). It assigned resources — construction, research, and upkeep capacities — to goals, attempting to maximize the total priority of the goals that could be satisfied. The priorities were set by a large number of hand-tuned values, which could be swapped for a different set to give the AI different personalities (Dill 2006). Each priority value was modified based on relevant factors of the current situation, a goal commitment value (to prevent flip-flopping once a goal has been selected) and a random value (to reduce predictability). It was found that this not only created a fun, challenging opponent, but also made the AI easier to update for changes in game design throughout the development process (Dill 2006).

Case-Based Planning

CBP is a planning technique that finds similar past situations from which to draw potential solutions to the current situation. In the case of a CBP system, the solutions found are a set of potential plans or subplans that are likely to be effective in the current situation. CBP systems can exhibit poor reactivity at the strategic level and excessive reactivity at the action level, not reacting to high-level changes in situation until a low-level action fails, or discarding an entire plan because a single action failed (Palma et al. 2011).

One of the first applications of CBP to RTS games was by Aha, Molineaux, and Ponsen (2005), who created a system that extended the dynamic scripting concept of Ponsen et al. (2005) to select tactics and strategy based on the current situation. Using this technique, their system was able to play against a nonstatic opponent instead of requiring additional training each time the opponent changed. They reduced the complexity of the state and action spaces by abstracting states into a state lattice of possible orders in which buildings are constructed in a game (build orders) combined with a small set of features, and abstracting actions into a set of tactics generated for each state. This allowed their system to improve its estimate of the performance of each tactic in each situation over multiple games, and eventually learn to consistently beat all of the tested opponent scripts (Aha, Molineaux, and Ponsen 2005).

Ontañón et al. (2007) use the ideas of behaviors, goals, and alive-conditions from A Behavior Language (ABL, introduced by Mateas and Stern [2002]) combined with the ideas from earlier CBP systems to form a case-based system for playing Wargus. The cases are learned from human-annotated game logs, with each case detailing the goals a human was attempting to achieve with particular sequences of actions in a particular state. These cases can then be adapted and applied in-game to attempt to change the game state. By reasoning about a tree of goals and subgoals to be completed, cases can be selected and linked together into plan to satisfy the overall goal of winning the game (figure 5).

During the execution of a plan, it may be modified in order to adapt for unforeseen events or compensate for a failure to achieve a goal.

Mishra, Ontañón, and Ram (2008) extend the work of Ontañón et al. (2007) by adding a decision tree model to provide faster and more effective case retrieval. The decision tree is used to predict a high-level situation, which determines the attributes and attribute weights to use for case selection. This helps by skipping unnecessary attribute calculations and comparisons, and emphasizing important attributes. The decision tree and weightings are learned from game logs that have been human annotated to show the high-level situation at each point throughout the games. This annotation increased the development effort required for the AI system but successfully provided better and faster case retrieval than the original system (Mishra, Ontañón, and Ram 2008).

More recent work using CBP tends to focus on the learning aspects of the system instead of the planning aspects. As such, it is discussed further in the Plan Recognition and Learning section.

A different approach is taken by Cadena and Garrido (2011), who combine the ideas of CBR with those of fuzzy sets, allowing the reasoner to abstract state information by grouping continuous feature values. This allows them to vastly simplify the state space, and it may be a closer representation of human thinking, but could potentially result in the loss of important information. For strategic decision making, their system uses regular cases made up of exact unit and building counts, and selects a plan made up of five high-level actions, such as creating units or buildings. But for tactical reasoning (micromanagement is not explored), their system maintains independent fuzzy state descriptions and carries out independent CBR for each region of the map, thus avoiding reasoning about the map as a whole at the tactical level. Each region's state includes a linguistic fuzzy representation of its area (for example, small, medium, big), choke points, military presence, combat intensity, lost units, and

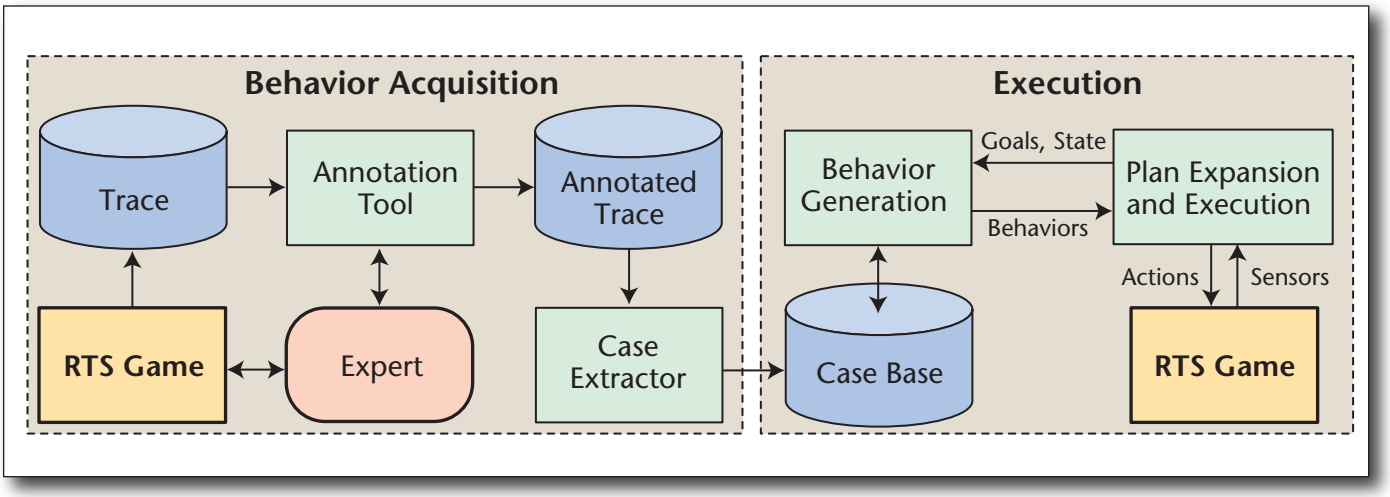


Figure 5. A Case-Based Planning Approach.

The approach uses cases of actions extracted from annotated game logs to form plans that satisfy goals in Wargus. Adapted from Ontañón et al. (2007).

amounts of each friendly and enemy unit type (for example, none, few, many). After building the case base from just one replay of a human playing against the in-built AI, the system was able to win around 60 percent of games (and tie in about 15 percent) against the AI on the same map. However, it is unclear how well the system would fare at the task of playing against different races (unique playable teams) and strategies, or playing on different maps.

Hierarchical Planning

By breaking up a problem hierarchically, planning systems are able to deal with parts of the situation separately at different levels of abstraction, reducing the complexity of the problem, but creating a potential new issue in coordination between the different levels (Marthi et al. 2005; Weber et al. 2010). A hierarchical plan maps well to the hierarchy of goals and subgoals typical in RTS games, from the highest-level goals such as winning the game, to the lowest-level goals, which map directly to in-game actions. Some researchers formalize this hierarchy into the well-defined structure of a hierarchical task network (HTN), which contains tasks, their ordering, and methods for achieving them. High-level, complex tasks in an HTN may be decomposed into a sequence of simpler tasks, which themselves can be decomposed until each task represents a concrete action (Muñoz-Avila and Aha 2004).

HTNs have been used for strategic decision making in RTS games, but not for StarCraft. Muñoz-Avila and Aha (2004) focus on the explanations that an HTN planner is able to provide to a human querying its behavior, or the reasons underlying certain events, in the context of an RTS game. Laagland (2008) implements and tests an agent capable of playing an open source RTS called Spring[11] using a hand-crafted HTN. The HTN allows the agent to react dynamically to problems, such as rebuilding a building that is lost or gathering additional resources of a particular type when needed, unlike the built-in scripted AI. Using a balanced strategy, the HTN agent usually beats the built-in AI in Spring, largely due to better resource management. Efforts to learn HTNs, such as Nejati, Langley, and Konik (2006), have been pursued in much simpler domains, but never directly used in the field of RTS AI. This area may hold promise in the future for reducing the work required to build HTNs.

An alternative means of hierarchical planning was used by Weber et al. (2010). They use an active behavior tree in A Behavior Language, which has parallel, sequential, and conditional behaviors and goals in a tree structure (figure 6) very similar to a behavior tree (discussed in the next subsection). However, in this model, the tree is expanded during execution by selecting behaviors (randomly, or based on conditions or priority) to satisfy goals, and different behaviors can communicate indirectly by reading or writing information on a shared whiteboard.

Hierarchical planning is often combined as part of other methods, such as how Ontañón et al. (2007) use a hierarchical CBP system to reason about goals and plans at different levels.

Behavior Trees

Behavior trees are hierarchies of decision and action nodes that are commonly used by programmers and designers in the game industry in order to define behaviors (effectively a partial plan) for agents (Palma et al. 2011). They have become popular because, unlike scripts, they can be created and edited using visual tools, making them much more accessible and understandable to nonprogrammers (Palma et al.

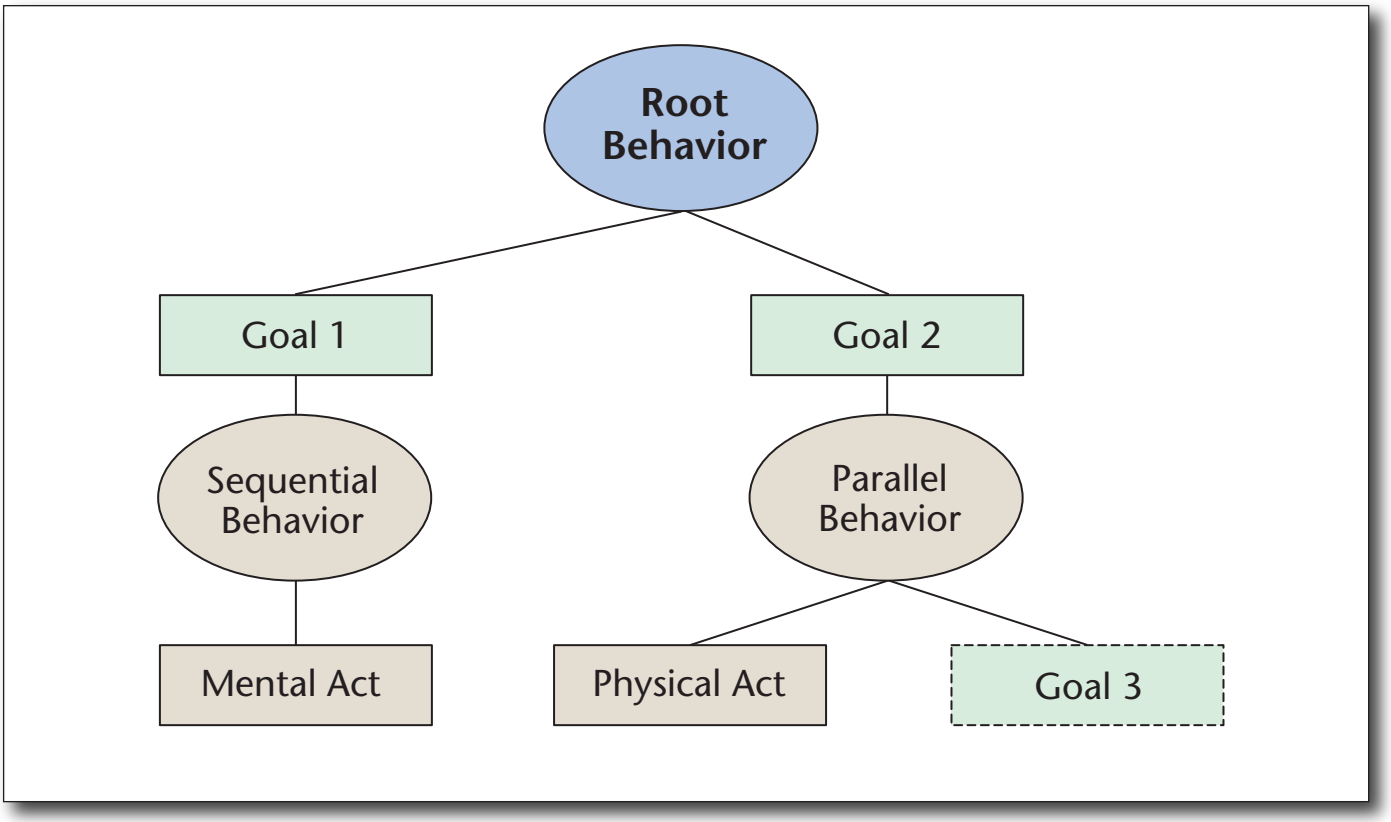


Figure 6. A Simple Active Behavior Tree Used for Hierarchical Planning.

The figure shows mental acts (calculation or processing), physical acts (in-game actions), and an unexpanded goal. Adapted from Weber et al. (2010).

2011). Additionally, their hierarchical structure encourages reuse, as a tree defining a specific behavior can be attached to another tree in multiple positions or can be customized incrementally by adding nodes (Palma et al. 2011). Because behavior trees are hierarchical, they can cover a wide range of behavior, from very low-level actions to strategic-level decisions. Palma et al. (2011) use behavior trees to enable direct control of a case-based planner's behavior. With their system, machine learning can be used to create complex and robust behavior through the planner, while allowing game designers to change specific parts of the behavior by substituting a behavior tree instead of an action or a whole plan. This means they can define custom behavior for specific scenarios, fix incorrectly learned behavior, or tweak the learned behavior as needed.

Goal-Driven Autonomy

GDA is a model in which "an agent reasons about its goals, identifies when they need to be updated, and changes or adds to them as needed for subsequent planning and execution" (Molineaux, Klenk, and Aha 2010). This addresses the high- and low-level reactivity problem experienced by CBP by actively reasoning about and reacting to why a goal is succeeding or failing.

Weber, Mateas, and Jhala (2010) describe a GDA system for StarCraft using A Behavior Language, which is able to form plans with expectations about the outcome. If an unexpected situation or event occurs, the system can record it as a discrepancy, generate an explanation for why it occurred, and form new goals to revise the plan, allowing the system to react appropriately to unforeseen events (figure 7). It is also capable of simultaneously reasoning about multiple goals at differing granularity. It was initially unable to learn goals, expectations, or strategies, so this knowledge had to be input and updated manually, but later improvements allowed these to be learned from demonstration (discussed in the next section) (Weber, Mateas, and Jhala 2012). This system was used in the Artificial Intelligence and Interactive Digital Entertainment (AIIDE) StarCraft AI competition entry EISBot and was also evaluated by playing against human players on a competitive StarCraft ladder called International Cyber Cup (ICCup),[12] where players are ranked based on their performance — it attained a ranking indicating it was better than 48 percent of the competitive play-

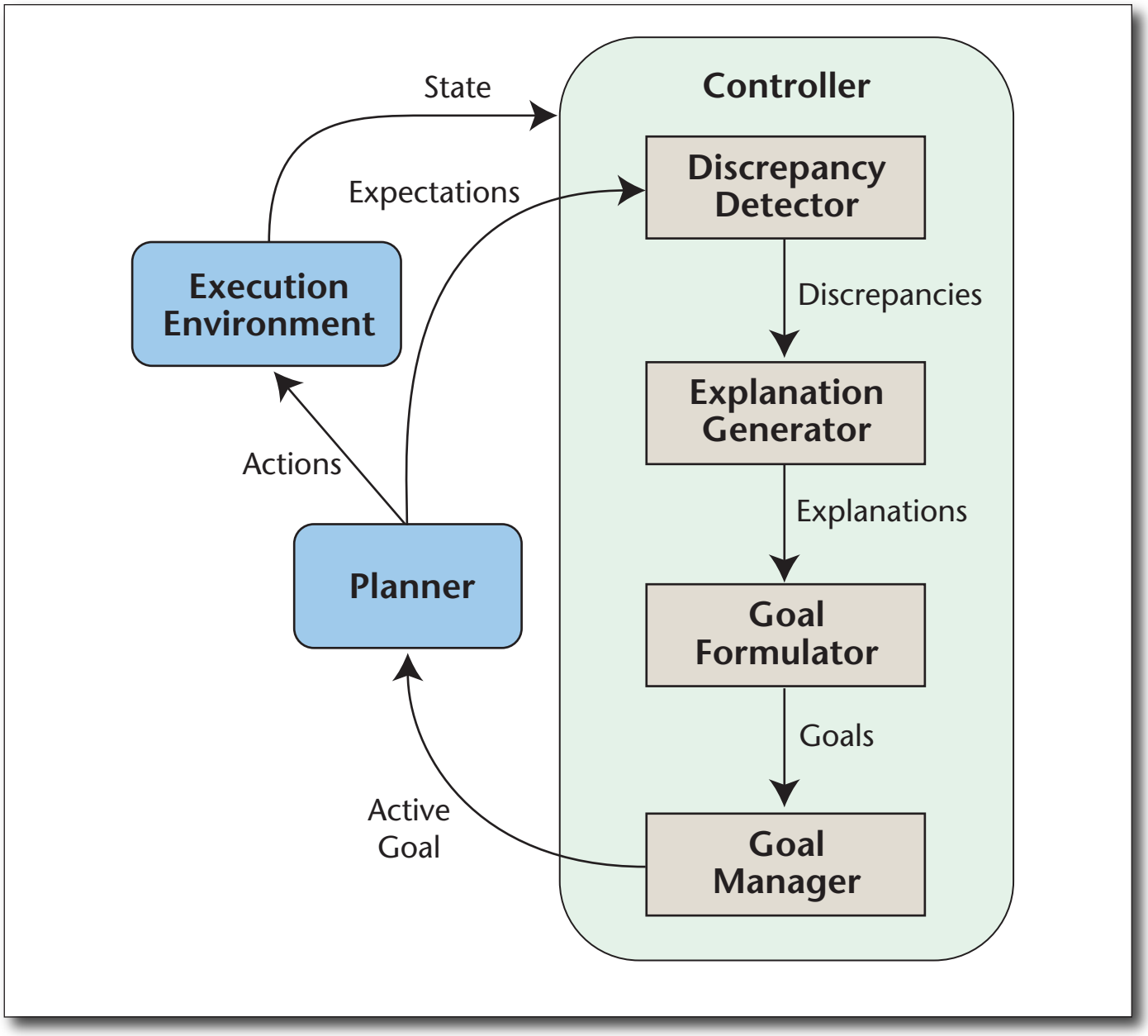


Figure 7. GDA Conceptual Model.

A planner produces actions and expectations from goals, and unexpected outcomes result in additional goals being produced (Weber, Mateas, and Jhala 2012).

ers (Weber, Mateas, and Jhala 2010; Weber et al. 2010).

Jaidee, Muñoz-Avila, and Aha (2011) integrate CBR and RL to make a learning version of GDA, allowing their system to improve its goals and domain knowledge over time. This means that less work is required from human experts to specify possible goals, states, and other domain knowledge because missing knowledge can be learned automatically. Similarly, if the underlying domain changes, the learning system is able to adapt to the changes automatically. However, when applied to a simple domain, the system was unable to beat the performance of a nonlearning GDA agent (Jaidee, Muñoz-Avila, and Aha 2011).

State Space Planning

Automated planning and scheduling is a branch of classic AI research from which heuristic state space planning techniques have been adapted for planning in RTS game AI. In these problems, an agent is given a start and goal state, and a set of actions that have preconditions and effects. The agent must then find a sequence of actions to achieve the goal from

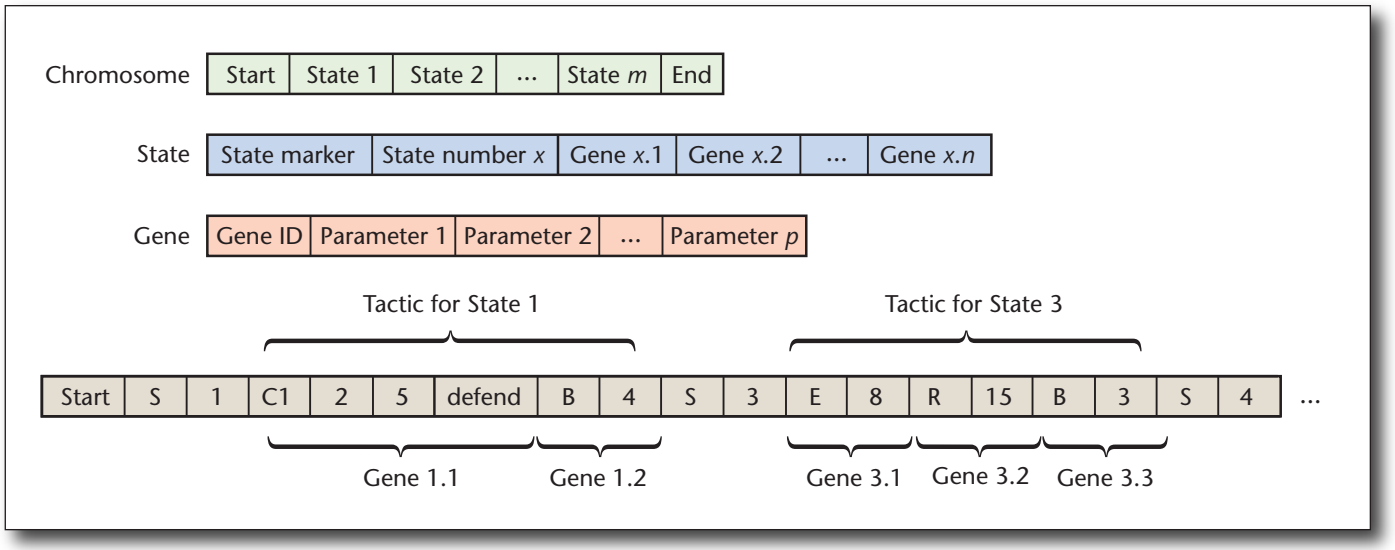


Figure 8. Design of a Chromosome for Evolving RTS Game AI Strategies.

(Ponsen et al. 2005)

the starting state. Existing RTS applications add complexity to the basic problem by dealing with durative and parallel actions, integer-valued state variables, and tight time constraints.

Automated planning ideas have already been applied successfully to commercial first-person shooter (FPS) games within an architecture called Goal-Oriented Action Planning (GOAP). GOAP allows agents automatically to select the most appropriate actions for their current situation in order to satisfy a set of goals, ideally resulting in more varied, complex, and interesting behavior, while keeping code more reusable and maintainable (Orkin 2004). However, GOAP requires a large amount of domain engineering to implement and is limited because it maps states to goals instead of to actions, so the planner cannot tell whether achieving goals is going according to the plan, failing, or has failed (Orkin 2004; Weber, Mateas, and Jhala 2010). Furthermore, Champandard[13] states that GOAP has now turned out to be a dead end, as academe and industry have moved away from GOAP in favor of hierarchical planners to achieve better performance and code maintainability.

However, Chan et al. (2007) and Churchill and Buro (2011) use an automated planning-based approach similar to GOAP to plan build orders in RTS games. Unlike GOAP, they are able to focus on a single goal: finding a plan to build a desired set of units and buildings in a minimum duration (makespan). The RTS domain is simplified by abstracting resource collection to an income rate per worker, assuming building placement and unit movement takes a constant amount of time, and completely ignoring opponents. Ignoring opponents is fairly reasonable for the beginning of a game, as there is generally little opponent interaction, and doing so means the planner does not have to deal with uncertainty and external influences on the state. Both of these methods still require expert knowledge to provide a goal state for them to pursue.

The earlier work by Chan et al. (2007) uses a combination of means-ends analysis and heuristic scheduling in Wargus. Means-ends analysis produces a plan with a minimal number of actions required to achieve the goal, but this plan usually has a poor makespan because it doesn't consider concurrent actions or actions that produce greater resources. A heuristic scheduler then reorganizes actions in the plan to start each action as soon as possible, adding concurrency and reducing the makespan. To consider producing additional resources, the same process is repeated with an extra goal for producing more of a resource (for each resource) at the beginning of the plan, and the plan with the shortest makespan is used. The resulting plans, though nonoptimal, were found to be similar in length to plans executed by an expert player, and vastly better than plans generated by state-of-the-art general purpose planners (Chan et al. 2007).

Churchill and Buro (2011) improve upon the earlier work by using a branch-and-bound depth-first search to find optimal build orders within an abstracted simulation of StarCraft. In addition to the simplifications mentioned above, they avoid simulating individual time steps by allowing any action that will eventually complete without further player interaction, and jumping directly to the point at which each action completes for the next decision node. Even so, other smaller optimizations were needed to speed up the planning process enough to use in-game. The search used either the gathering

time or the build time required to reach the goal (whichever was longer) as the lower bound, and a random path to the goal as the upper bound (Churchill and Buro 2011). The system was evaluated against professional build orders seen in replays, using the set of units and buildings owned by the player at a particular time as the goal state. Due to the computational cost of planning later in the game, planning was restricted to 120 seconds ahead, with replanning every 30 seconds. This produced shorter or equal-length plans to the human players at the start of a game, and similar-length plans on average (with a larger variance) later in the game. It remains to be seen how well this method would perform for later stages of the game, as only the first 500 seconds were evaluated and searching took significantly longer in the latter half. However, this appears to be an effective way to produce near-optimal build orders for at least the early to middle game of StarCraft (Churchill and Buro 2011).

Evolutionary Algorithms

Evolutionary algorithms search for an effective solution to a problem by evaluating different potential solutions and combining or randomizing components of high-fitness potential solutions to find new, better solutions. This approach is used infrequently in the RTS Game AI field, but it has been effectively applied to the subproblem of tactical decision making in StarCraft (discussed earlier) and learning strategic knowledge in similar RTS titles.

Although evolutionary algorithms have not yet been applied to strategic decision making in StarCraft, they have been applied to its sequel, StarCraft II.[14] The Evolution Chamber[15] software uses the technique to optimize partially defined build orders. Given a target set of units, buildings, and upgrades to be produced by certain times in the match, the software searches for the fastest or least resource-intensive way of reaching these targets. Although there have not been any academic publications regarding this software, it gained attention by producing an unusual and highly effective plan in the early days of StarCraft II.

Ponsen et al. (2005) use evolutionary algorithms to generate strategies in a game of Wargus. To generate the strategies, the evolutionary algorithm combines and mutates sequences of tactical and strategic-level actions in the game to form scripts (figure 8) that defeat a set of human-made and previously evolved scripts. The fitness of each potential script is evaluated by playing it against the predefined scripts and using the resulting in-game military score combined with a time factor that favors quick wins or slow losses. Tactics are extracted as sequences of actions from the best scripts, and are finally used in a dynamic script that chooses particular tactics to use in a given state, based on its experience of their effectiveness — a form of RL. The resulting dynamic scripts are able to consistently beat most of the static scripts they were tested against after learning for approximately 15 games against that opponent, but were unable to consistently beat some scripts after more than 100 games (Ponsen et al. 2005; 2006). A drawback of this method is that the effectiveness values learned for the dynamic scripts assume that the opponent is static and would not adapt well to a dynamic opponent (Aha, Molineaux, and Ponsen 2005).

Cognitive Architectures

An alternative method for approaching strategic-level RTS game AI is to model a reasoning mechanism on how humans are thought to operate. This could potentially lead toward greater understanding of how humans reason and allow us to create more human-like AI. This approach has been applied to StarCraft as part of a project using the Soar cognitive architecture, which adapts the BWAPI interface to communicate with a Soar agent.[16] It makes use of Soar's spatial visual system to deal with reconnaissance activities and pathfinding, and Soar's working memory to hold perceived and reasoned state information. However, it is currently limited to playing a partial game of StarCraft, using only the basic barracks and marine units for combat, and using hard-coded locations for building placement.[16]

A similar approach was taken by Wintermute, Xu, and Laird (2007), but it applied Soar to ORTS instead of StarCraft. They were able to interface the Soar cognitive architecture to ORTS by reducing the complexity of the problem using the concepts of grouping and attention for abstraction. These concepts are based on human perception, allowing the underlying Soar agent to receive information as a human would, postperception — in terms of aggregated and filtered information. The agent could view entire armies of units as a single entity, but could change the focus of its attention, allowing it to perceive individual units in one location at a time, or groups of units over a wide area (figure 9). This allowed the agent to control a simple strategic-level RTS battle situation without being overwhelmed by the large number of units (Wintermute, Xu, and Laird 2007). However, due to the limitations of Soar, the agent could pursue only one goal at a time, which would be very limiting in StarCraft and most complete RTS games.

Spatial Reasoning

RTS AI agents have to be able to reason about the positions and actions of often large numbers of hidden objects, many with different properties, moving over time, controlled by an opponent in a dynamic environment (Weber, Mateas, and Jhala 2011b; Wintermute, Xu, and Laird 2007). Despite the complexity of the problem, humans can reason about this information very quickly and accurately, often predicting and intercepting the location of an enemy

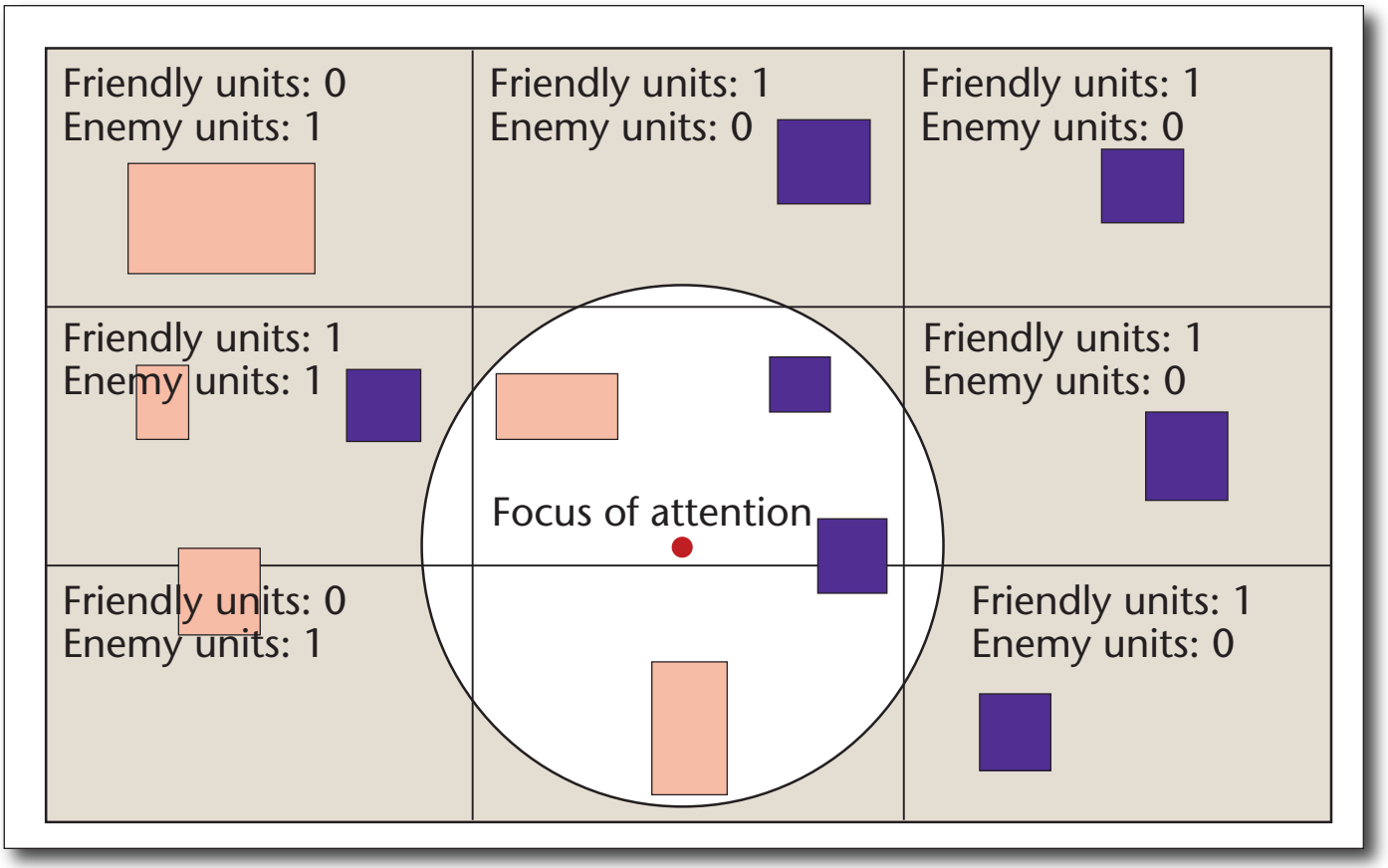


Figure 9. Attention Limits the Information the Agent Receives by Hiding or Abstracting Objects Further from the Agent's Area of Focus.

(Wintermute, Xu, and Laird 2007).

attack or escape based on very little information, or using terrain features and the arrangement of their own units and buildings to their advantage. This makes RTS a highly suitable domain for spatial reasoning research in a controlled environment (Buro 2004; Weber, Mateas, and Jhala 2011a; Wintermute, Xu, and Laird 2007).

Even the analysis of the terrain in RTS games, ignoring units and buildings, is a nontrivial task. In order to play effectively, players need to be able to know which regions of the terrain are connected to other regions, and where and how these regions connect. The connections between regions are as important as the regions themselves, because they offer defensive positions through which an army must move to get into or out of the region (choke points). Perkins (2010) describes the implementation and testing of the Brood War Terrain Analyzer, which has become a very common library for creating StarCraft bots capable of reasoning about their terrain. The library creates and prunes a Voronoi diagram using information about the walkable tiles of the map, identifies nodes as regions or choke points, then merges adjacent regions according to thresholds that were determined by trial and error to produce the desired results. The choke point nodes are converted into lines that separate the regions, resulting in a set of region polygons connected by choke points (figure 10). When compared against the choke points identified by humans, it had a 0–17 percent false negative rate, and a 4–55 percent false positive rate, and took up to 43 seconds to analyze the map, so there is still definite room for improvement (Perkins 2010).

Once a player is capable of simple reasoning about the terrain, it is possible to begin reasoning about the movement of units over this terrain. A particularly useful spatial reasoning ability in RTS games is to be able to predict the location of enemy units while they are not visible to a player. Weber, Mateas, and Jhala (2011b) use a particle model for predicting enemy unit positions in StarCraft, based on the unit's trajectory and nearby choke points at the time it was seen. A single particle was used for each unit instead of a particle cloud because it is not possible to visually distinguish between two units of the same type, so it would be difficult to update the cloud if a unit was lost then resighted (Weber, Mateas, and Jhala 2011b). In order to account for the differences between the unit types in StarCraft, they divided the

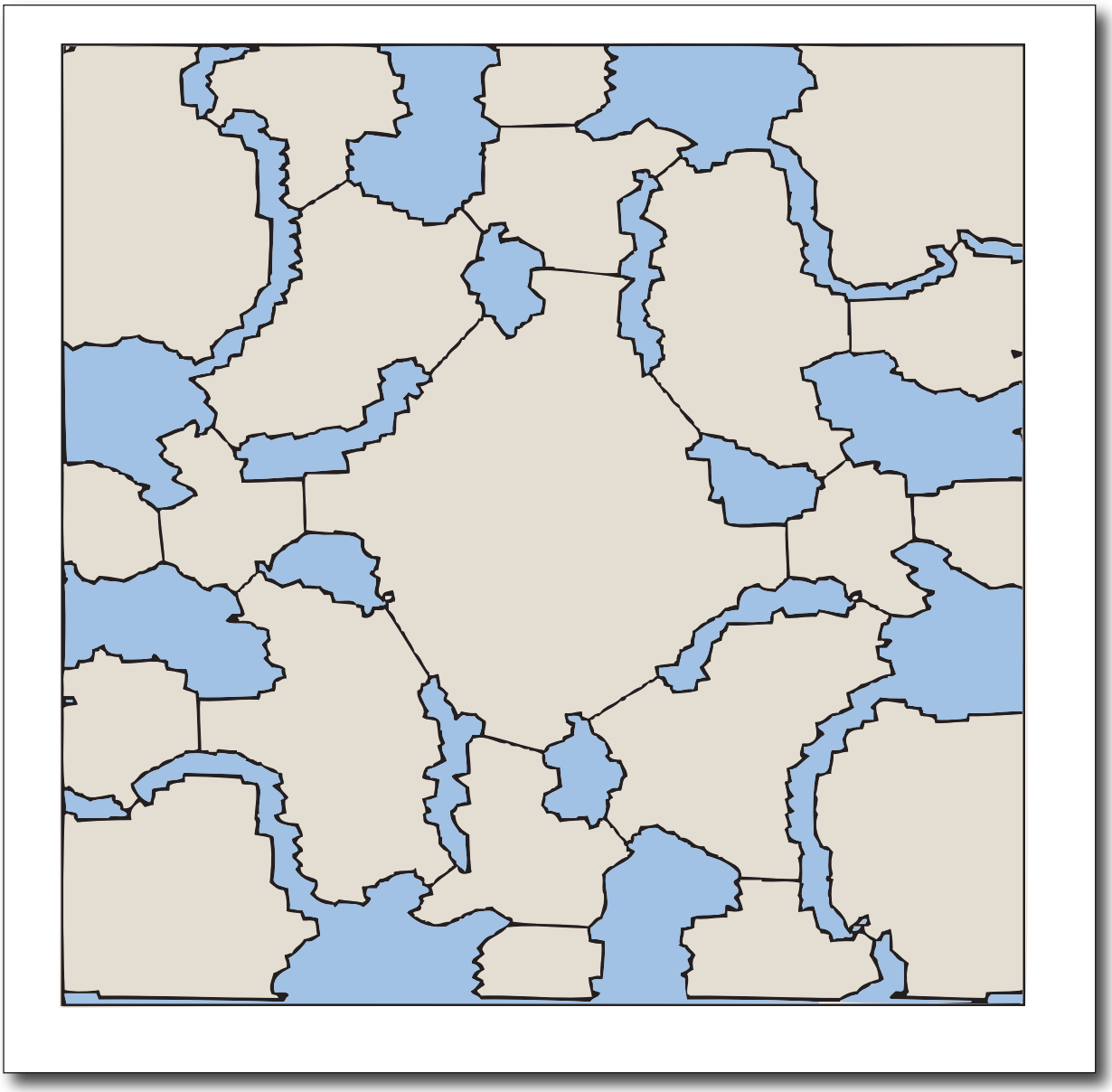


Figure 10. Terrain After Analysis.

The figure shows impassable areas in blue and choke points as lines between light areas (Perkins 2010).

types into broad classes and learned a movement model for each class from professional replays on a variety of maps. The model allowed their bot to predict, with decreasing confidence over time, the subsequent locations of enemy units after sighting them, resulting in an increased win rate against other bots (Weber, Mateas, and Jhala 2011b).

The bulk of spatial reasoning research in StarCraft and other RTS games is based on potential fields (PFs), and to a lesser extent, influence maps. Each of these techniques help to aggregate and abstract spatial information by summing the effect of individual points of information into a field over an area, allowing decisions to be made based on the computed field strength at particular positions. They were first applied to RTS games by Hagelbäck and Johansson (2008), before which they were used for robot navigation. Kabanza et al. (2010) use an influence map to evaluate the potential threats and opportunities of an enemy force in an effort to predict the opponent's strategy, and Uriarte and Ontañón (2012) use one to evaluate threats and obstacles in order to control the movement of units performing a hit-and-run behavior known as kiting. Baumgarten, Colton, and Morris (2009) use a few different influence maps for synchronizing attacks by groups of units, moving and

grouping units, and choosing targets to attack. Weber and Ontañón (2010) use PFs to aid a CBP system by taking the field strengths of many different fields at a particular position, so that the position is represented as a vector of field strengths, and can be easily compared to others stored in the case base. Synnaeve and Bessière (2011b) claim that their Bayesian model for unit movement subsumes PFs, as each unit is controlled by Bayesian sensory inputs that are capable of representing threats and opportunities in different directions relative to the unit. However, their system still needs to use damage maps in order to summarize this information for use by the sensory inputs (Synnaeve and Bessière 2011b).

PFs were used extensively in the Overmind StarCraft bot, for both offensive and defensive unit behavior (Huang 2011). The bot used the fields to represent opportunities and threats represented by known enemy units, using information about unit statistics so that the system could estimate how beneficial and how costly it would be to attack each target. This allowed attacking units to treat the fields as attractive and repulsive forces for movement, resulting in them automatically congregating on high-value targets and avoiding defenses.

Additionally, the PFs were combined with temporal reasoning components, allowing the bot to consider the time cost of reaching a faraway target, and the possible movement of enemy units around the map, based on their speed and visibility. The resulting threat map was used for threat-aware pathfinding, which routed units around more threatening regions of the map by giving movement in threatened areas a higher path cost. The major difficulty they experienced in using PFs so much was in tuning the strengths of the fields, requiring them to train the agent in small battle scenarios in order to find appropriate values (Huang 2011). To the authors' knowledge, this is the most sophisticated spatial reasoning that has been applied to playing StarCraft.

Plan Recognition and Learning

A major area of research in the RTS game AI literature involves learning effective strategic-level game play. By using an AI system capable of learning strategies, researchers aim to make computer opponents more challenging, dynamic, and humanlike, while making them easier to create (Hsieh and Sun 2008). StarCraft is a very complex domain to learn from, so it may provide insights into learning to solve real-world problems. Some researchers have focused on the subproblem of determining an opponent's strategy, which is particularly difficult in RTS games due to incomplete information about the opponent's actions, hidden by the "fog of war" (Kabanza et al. 2010). Most plan recognition makes use of an existing plan library to match against when attempting to recognize a strategy, but some methods allow for plan recognition without any predefined plans (Cheng and Thawonmas 2004; Synnaeve and Bessière 2011a). Often, data is extracted from the widely available replays files of expert human players, so a data set was created in order to reduce repeated work (Synnaeve and Bessière 2012). This section divides the plan recognition and learning methods into deductive, abductive, probabilistic, and case-based techniques. Within each technique, plan recognition can be either intended — plans are denoted for the learner and there is often interaction between the expert and the learner — or keyhole — plans are indirectly observed and there is no two-way interaction between the expert and the learner.

Deductive

Deductive plan recognition identifies a plan by comparing the situation with hypotheses of expected behavior for various known plans. By observing particular behavior a deduction can be made about the plan being undertaken, even if complete knowledge is not available. The system described by Kabanza et al. (2010) performs intended deductive plan recognition in StarCraft by matching observations of its opponent against all known strategies that could have produced the situation. It then simulates the possible plans to determine expected future actions of its opponent, judging the probability of plans based on new observations and discarding plans that do not match (figure 11). The method used requires significant human effort to describe all possible plans in a decision tree type structure (Kabanza et al. 2010).

The decision tree machine learning method used by Weber and Mateas (2009) is another example of intended deductive plan recognition. Using training data of building construction orders and timings that have been extracted from a large selection of StarCraft replay files, it creates a decision tree to predict which midgame strategy is being demonstrated. The replays are automatically given their correct classification through a rule set based upon the build order. The learning process was also carried out with a nearest neighbor algorithm and a nonnested generalized exemplars algorithm. The resulting models were then able to predict the build order from incomplete information, with the nearest neighbor algorithm being most robust to incomplete information (Weber and Mateas 2009).

Abductive

Abductive plan recognition identifies plans by making assumptions about the situation that are sufficient to explain the observations. The GDA system described by Weber, Mateas, and Jhala (2010) is an example of intended abductive plan recognition in StarCraft, where expectations are formed about the result of actions, and unexpected events are accounted for as discrepancies. The planner handles discrepancies by choosing from a set of predefined explana-

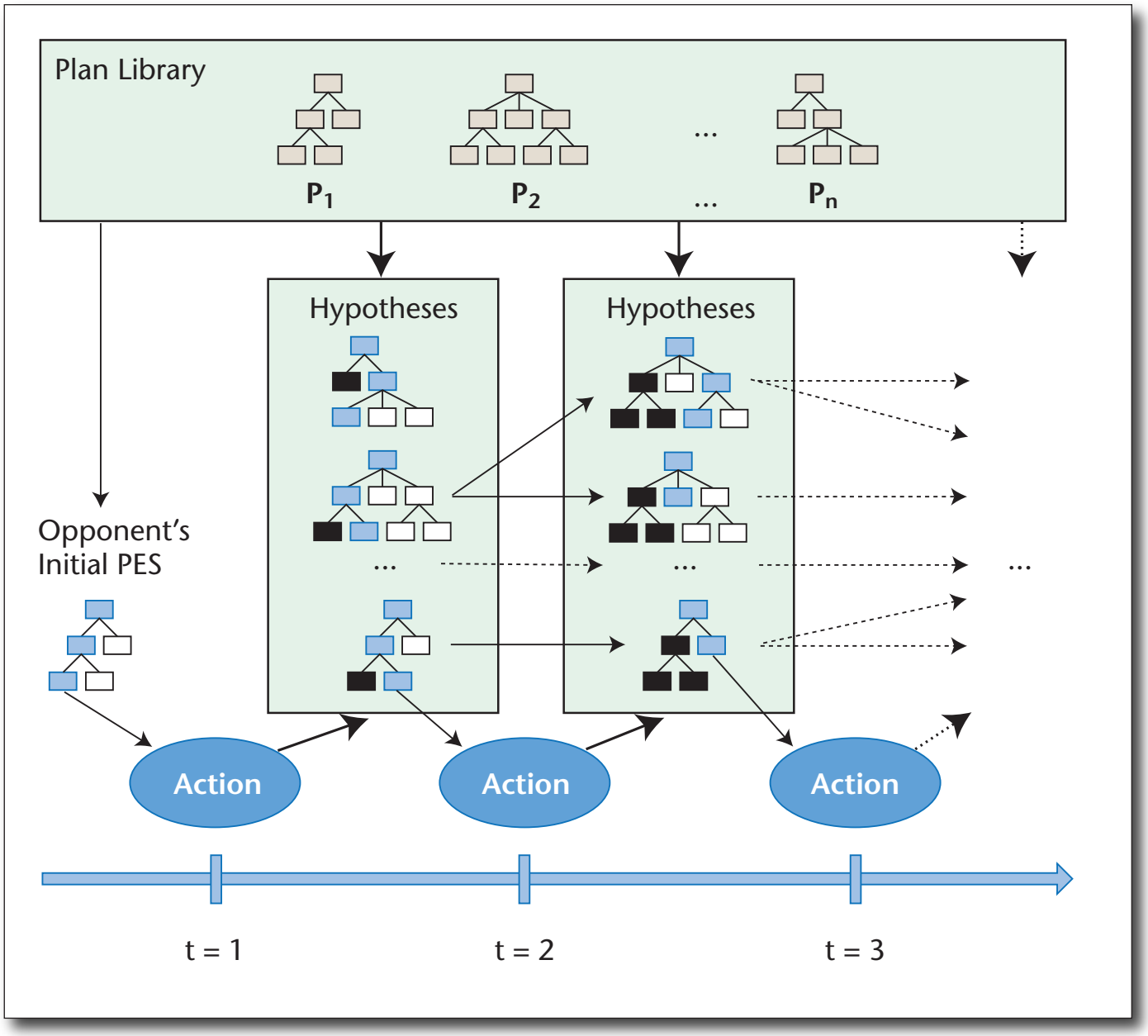


Figure 11. New Observations Update an Opponent's Possible Plan Execution Statuses to Determine Which Plans Are Potentially Being Followed.

(Kabanza et al. 2010).

tions that give possible reasons for discrepancies and create new goals to compensate for the change in assumed situation. This system required substantial domain engineering in order to define all of the possible goals, expectations, and explanations necessary for a domain as complex as StarCraft.

Later work added the ability for the GDA system to learn domain knowledge for StarCraft by analyzing replays offline (Weber, Mateas, and Jhala 2012). In this modified system, a case library of sequential game states was built from the replays, with each case representing the player and opponent states as numerical feature vectors. Then case-based goal formulation was used to produce goals at run time. The system forms predictions of the opponent's future state (referred to as explanations in the article) by finding a similar opponent state to the current opponent state in the case library, looking at the future of the similar state to find the difference in the feature vectors over a set period of time, and then applying this difference to the current opponent state to produce an expected opponent state. In a similar manner, it produces a goal state by finding the expected future player state, using the predicted opponent

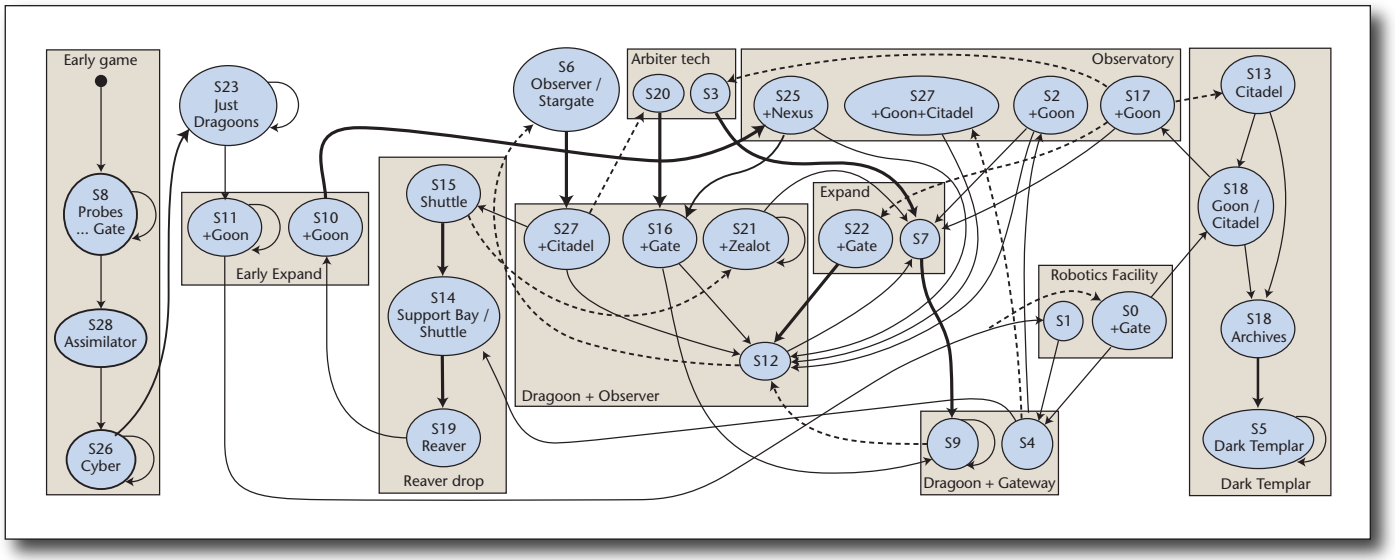


Figure 12. State Transition Graph.

As learned in Dereszynski et al. (2011), showing transitions with probability at least 0.25 as solid edges, and higher-probability transitions with thicker edges. Dotted edges are low-probability transitions shown to make all nodes reachable. Labels in each state are likely units to be produced, while labels outside states are a human analysis of the strategy exhibited. (Dereszynski et al. 2011).

state instead of the current state in order to find appropriate reactions to the opponent. Expectations are also formed from the case library, using changes in the opponent state to make predictions about when new types of units will be produced. When an expectation is not met (within a certain tolerance for error), a discrepancy is created, triggering the system to formulate a new goal. The resulting system appeared to show better results in testing than the previous ones, but further testing is needed to determine how effectively it adapts to unexpected situations (Weber, Mateas, and Jhala 2012).

Probabilistic

Probabilistic plan recognition makes use of statistics and expected probabilities to determine the most likely future outcome of a given situation. Synnaeve and Bessière (2011a), Dereszynski et al. (2011), and Hostetler et al. (2012) carry out keyhole probabilistic plan recognition in StarCraft by examining build orders from professional replays, without any prior knowledge of StarCraft build orders. This means they should require minimal work to adapt to changes in the game or to apply to a new situation, because they can learn directly from replays without any human input. The models learned can then be used to predict unobserved parts of the opponent's current state, or the future strategic direction of a player, given the player's current and past situations. Alternatively, they can be used to recognize an unusual strategy being used in a game. The two approaches differ in the probabilistic techniques that are used, the scope in which they are applied, and the resulting predictive capabilities of the systems.

Dereszynski et al. (2011) use hidden Markov models to model the player as progressing through a series of states, each of which has probabilities for producing each unit and building type, and probabilities for which state will be transitioned to next. The model is applied to one of the sides in just one of the six possible race matchups, and to only the first seven minutes of game play, because strategies are less dependent on the opponent at the start of the game. State transitions happen every 30 seconds, so the timing of predicted future events can be easily found, but it is too coarse to capture the more frequent events, such as building new worker units. Without any prior information, it is able to learn a state transition graph that closely resembles the commonly used opening build orders (figure 12), but a thorough analysis and evaluation of its predictive power is not provided (Dereszynski et al. 2011).

Hostetler et al. (2012) extend previous work by Dereszynski et al. (2011) using a dynamic Bayesian network model for identifying strategies in StarCraft. This model explicitly takes into account the reconnaissance effort made by the player — measured by the proportion of the opponent's main base that has been seen — in order to determine whether a unit or building was not seen because it was not present, or because little effort was made to find it. This means that failing to find a unit can actually be very informative, provided enough effort was made. The model is also more precise than prior work, predicting exact counts and production of each unit and building type each 30-second time period, instead of just presence or absence. Production of units and buildings each time period is dependent on the current state,

based on a hidden Markov model as in Dereszynski et al. (2011). Again, the model was trained and applied to one side in one race matchup, and results are shown for just the first seven minutes of game play. For predicting unit quantities, it outperforms a baseline predictor, which simply predicts the average for the given time period, but only after reconnaissance has begun. This highlights a limitation of the model: it cannot differentiate easily between sequential time periods with similar observations, and therefore has difficulty making accurate predictions for during and after such periods. This happens because the similar periods are modeled as a single state that has a high probability of transitioning to the same state in the next period. For predicting technology structures, the model seems to generally outperform the baseline, and in both prediction tasks it successfully incorporates negative information to infer the absence of units (Hostetler et al. 2012).

Synnaeve and Bessière (2011a) carry out a similar process using a Bayesian model instead of a hidden Markov model. When given a set of thousands of replays, the Bayesian model learns the probabilities of each observed set of buildings existing at one-second intervals throughout the game. These timings for each building set are modeled as normal distributions, such that few or widely spread observations will produce a large standard deviation, indicating uncertainty (Synnaeve and Bessière 2011a). Given a (partial) set of observations and a game time, the model can be queried for the probabilities of each possible building set being present at that time. Alternatively, given a sequence of times, the model can be queried for the most probable building sets over time, which can be used as a build order for the agent itself (Synnaeve and Bessière 2011a).

The model was evaluated and shown to be robust to missing information, producing a building set with a little over one building wrong, on average, when 80 percent of the observations were randomly removed. Without missing observations and allowing for one building wrong, it was able to predict almost four buildings into the future, on average (Synnaeve and Bessière 2011a).

Case Based

Case-based plan recognition may also be carried out using case-based reasoning as a basis. CBR works by storing cases that represent specific knowledge of a problem and solution, and comparing new problems to past cases in order to adapt and reuse past solutions (Aamodt and Plaza 1994). It is commonly used for learning strategic play in RTS games because it can capture complex, incomplete situational knowledge gained from specific experiences to attempt to generalize about a very large problem space, without the need to transform the data (Aamodt and Plaza 1994; Floyd and Esfandiari 2009; Sánchez-Pelegrín, Gómez-Martín, and Díaz-Agudo 2005).

Hsieh and Sun (2008) use CBR to perform keyhole recognition of build orders in StarCraft by analyzing replays of professional players, similar to Synnaeve and Bessière (2011a) above. Hsieh and Sun (2008) use the resulting case base to predict the performance of a build order by counting wins and losses seen in the professional replays, which allows the system to predict which build order is likely to be more successful in particular situations.

In RTS games, CBR is often used not only for plan recognition but also as part of a more general method for learning actions and the situations in which they should be applied. An area of growing interest for researchers involves learning to play RTS games from a demonstration of correct behavior. These learning from demonstration techniques often use CBR and CBP, but they are discussed in their own section, which follows.

Although much of the recent work using CBR for RTS games learns from demonstration, Baumgarten, Colton, and Morris (2009) use CBR directly without observing human play. Their system uses a set of metrics to measure performance, in order to learn to play the strategy game DEFCON[17] through an iterative process similar to RL. The system uses cases of past games played to simultaneously learn which strategic moves it should make as well as which moves its opponent is likely to make. It abstracts lower-level information about unit and structure positions by using influence maps for threats and opportunities in an area and by grouping units into fleets and metafleets. In order for it to make generalizations about the cases it has stored, it groups the cases similar to its current situation using a decision tree algorithm, splitting the cases into more or less successful games based on game score and hand-picked metrics. A path through the resulting decision tree is then used as a plan that is expected to result in a high-scoring game. Attribute values not specified by the selected plan are chosen at random, so the system tries different moves until an effective move is found. In this way, it can discover new plans from an initially empty case base.

Learning by Observation

For a domain as complex as RTS games, gathering and maintaining expert knowledge or learning it through trial and error can be a very difficult task, but games can provide simple access to (some of) this information through replays or traces. Most RTS games automatically create traces, recording the events within a game and the actions taken by the players throughout the game. By analyzing the traces, a system can learn from the human demonstration of correct behavior, instead of requiring programmers to specify its behavior manually. This learning solely by observing the expert's external behavior and environment is usually called learning by observation, but is also known as apprenticeship

learning, imitation learning, behavioral cloning, programming by demonstration, and even learning from demonstration (Ontañón, Montana, and Gonzalez 2011). These learning methods are analogous to the way humans are thought to accelerate learning through observing an expert and emulating their actions (Mehta et al. 2009).

Although the concept can be applied to other areas, learning by observation (as well as learning from demonstration, discussed in the next section) is particularly applicable for CBR systems. It can reduce or remove the need for a CBR system designer to extract knowledge from experts or think of potential cases and record them manually (Hsieh and Sun 2008; Mehta et al. 2009). The replays can be transformed into cases for a CBR system by examining the actions players take in response to situations and events, or to complete certain predefined tasks.

In order to test the effectiveness of different techniques for learning by observation, Floyd and Esfandiari (2009) compared CBR, decision trees, support vector machines, and naïve Bayes classifiers for a task based on RoboCup robot soccer.[18] In this task, classifiers were given the perceptions and actions of a set of RoboCup players and were required to imitate their behavior. There was particular difficulty in transforming the observations into a form usable by most of the the classifiers, as the robots had an incomplete view of the field, so there could be very few or many objects observed at a given time (Floyd and Esfandiari 2009). All of the classifiers besides k-nearest neighbor — the classifier commonly used for CBR — required single-valued features or fixed-size feature vectors, so the missing values were filled with a placeholder item in those classifiers in order to mimic the assumptions of k-nearest neighbor. Classification accuracy was measured using the *f*-measure, and results showed that the CBR approach outperformed all of the other learning mechanisms (Floyd and Esfandiari 2009). These challenges and results may explain why almost all research in learning by observation and learning from demonstration in the complex domain of RTS games uses CBR as a basis.

Bakkes, Spronck, and van den Herik (2011) describe a case-based learning by observation system that is customized to playing Spring RTS games at a strategic level (figure 13), while the tactical decision making is handled by a script. In addition to regular CBR, with cases extracted from replays, they record a fitness value with each state, so the system can intentionally select suboptimal strategies when it is winning in order to make the game more evenly matched and more fun to play. This requires a good fitness metric for the value of a state, which is difficult to create for an RTS. In order to play effectively, the system uses hand-tuned feature weights on a chosen set of features, and chooses actions that are known to be effective against its expected opponent. The opponent strategy model is found by comparing observed features of the opponent to those of opponents in its case base, which are linked to the games where they were encountered. In order to make case retrieval efficient for accessing online, the case base is clustered and indexed with a fitness metric while offline. After playing a game, the system can add the replay to its case base in order to improve its knowledge of the game and opponent. A system capable of controlled adaptation to its opponent like this could constitute an interesting AI player in a commercial game (Bakkes, Spronck, and van den Herik 2011).

Learning by observation also makes it possible to create a domain-independent system that can simply learn to associate sets of perceptions and actions, without knowing anything about their underlying meaning (Floyd and Esfandiari 2010; 2011a). However, without domain knowledge to guide decisions, learning the correct actions to take in a given situation is very difficult. To compensate, the system must process and analyze observed cases, using techniques like automated feature weighting and case clustering in order to express the relevant knowledge.

Floyd and Esfandiari (2011a) claim their system is capable of handling complex domains with partial information and nondeterminism, and show it to be somewhat effective at learning to play robot soccer and Tetris, but it has not yet been applied to a domain as complex as StarCraft. Their system has more recently been extended to be able to compare perceptions based on the entire sequence of perceptions — effectively a trace — so that it is not limited to purely reactive behavior (Floyd and Esfandiari 2011b). In the modified model, each perceived state contains a link to the previous state, so that when searching for similar states to the current state, the system can incrementally consider additional past states to narrow down a set of candidates. By also considering the similarity of actions contained in the candidate cases, the system can stop comparing past states when all of the candidate cases suggested a similar action, thereby minimizing wasted processing time. In an evaluation where the correct action was dependent on previous actions, the updated system produced a better result than the original, but it is still unable to imitate an agent whose actions are based on a hidden internal state (Floyd and Esfandiari 2011b).

Learning from Demonstration

Instead of learning purely from observing the traces of interaction of a player with a game, the traces may be annotated with extra information — often about the player's internal reasoning or intentions — making the demonstrations easier to learn from, and providing more control over the particular behaviors learned. Naturally, adding annotations by hand makes the demonstrations more time-consuming to author, but some techniques have been developed to

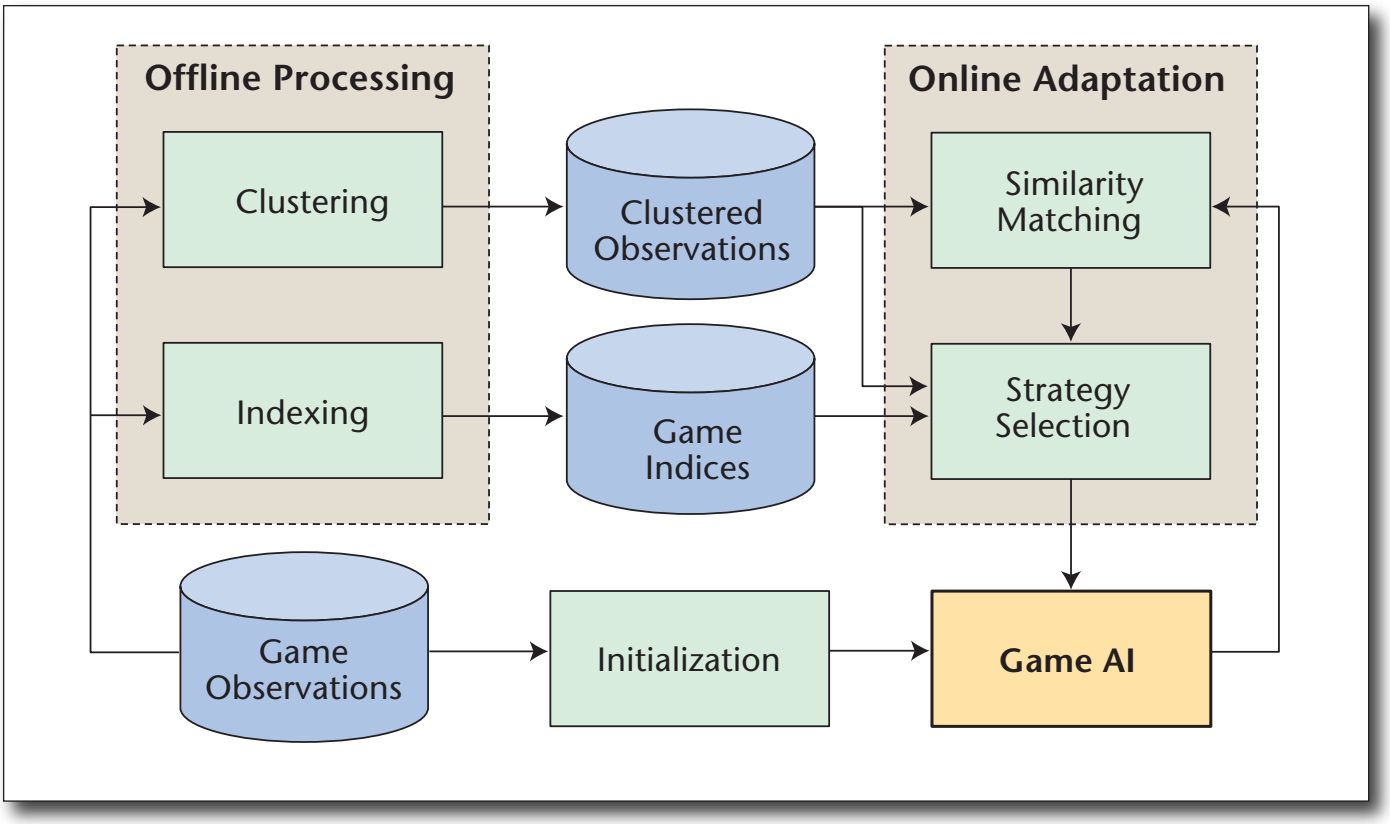


Figure 13. Learning by Observation Applied to an RTS.

Offline processing generalizes observations, initialization chooses an effective strategy, and online adaptation ensures cases are appropriate in the current situation. Adapted from Bakkes, Spronck, and van den Herik (2011).

automate this process. This method of learning from constructed examples is known as learning from demonstration.

Given some knowledge about the actions and tasks (things that we may want to complete) in a game, there are a variety of different methods that can be used to extract cases from a trace for use in learning by observation or learning from demonstration systems. Ontañón (2012) provides an overview of several different case acquisition techniques, from the most basic reactive and monolithic learning approaches to more complex dependency graph learning and time-span analysis techniques. Reactive learning selects a single action in response to the current situation, while monolithic sequential learning selects an entire game plan; the first has issues with preconditions and the sequence of actions, whereas the second has issues managing failures in its long-term plan (Ontañón 2012). Hierarchical sequential learning attempts to find a middle ground by learning which actions result in the completion of particular tasks, and which tasks' actions are subsets of other tasks' actions, making them subtasks. That way, ordering is retained, but when a plan fails it must only choose a new plan for its current task, instead of for the whole game (Ontañón 2012).

Sequential learning strategies can alternatively use dependency graph learning, which uses known preconditions and postconditions, and observed ordering of actions, to find a partial ordering of actions instead of using the total ordered sequence exactly as observed. However, these approaches to determining subtasks and dependencies produce more dependencies than really exist, because independent actions or tasks that coincidentally occur at a similar time will be considered dependent (Ontañón 2012). The surplus dependencies can be reduced using time-span analysis, which removes dependencies where the duration of the action indicates that the second action started before the first one finished. In an experimental evaluation against static AI, it was found that the dependency graph and time-span analysis improved the results of each strategy they were applied to, with the best results being produced by both techniques applied to the monolithic learning strategy (Ontañón 2012).

Mehta et al. (2009) describe a CBR and planning system that is able to learn to play the game Wargus from human-annotated replays of the game (figure 14). By annotating each replay with the goals that the player was trying to achieve at the time, the system can group sequences of actions into behaviors to achieve specific goals, and learn a hierarchy of goals and their possible orderings. The learned behaviors

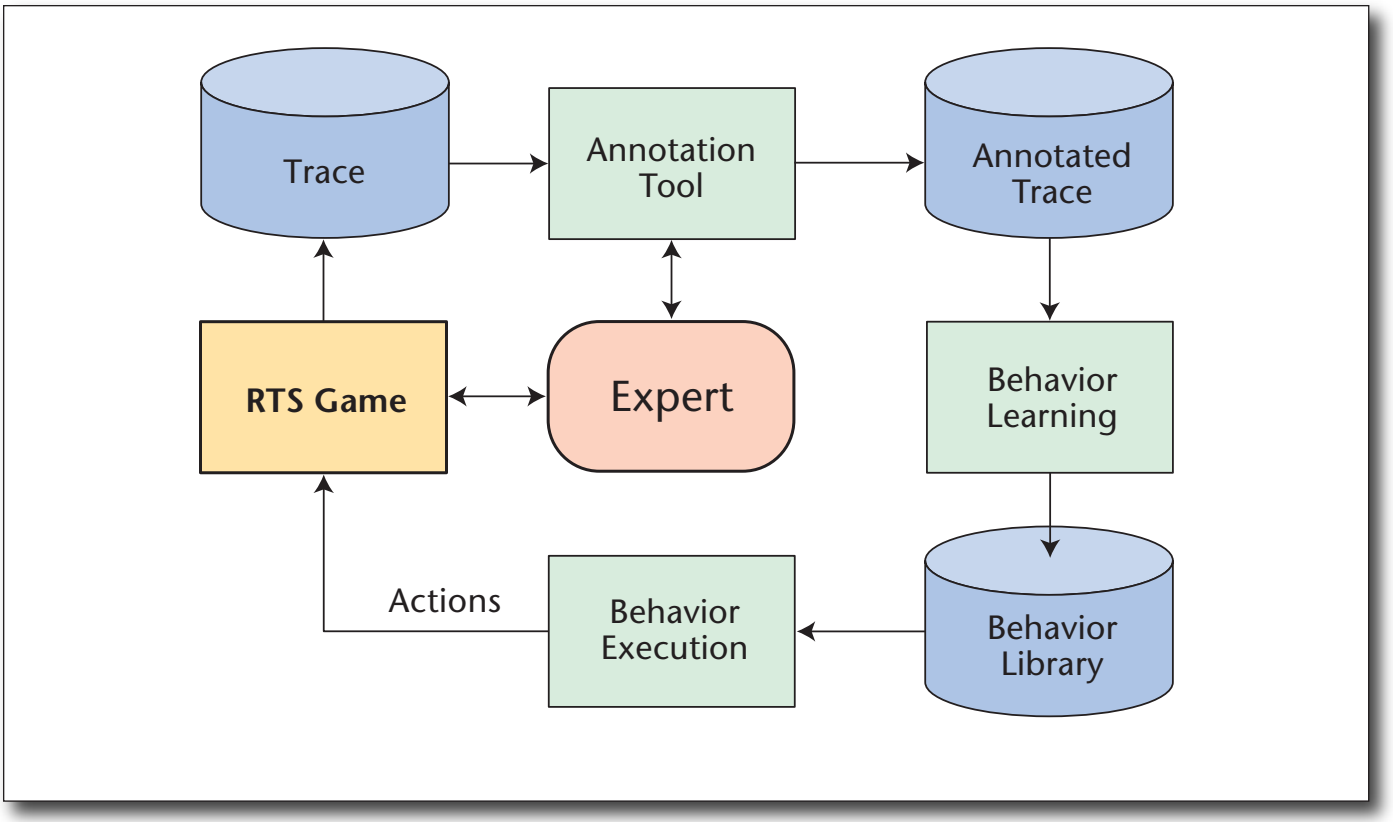


Figure 14. General Architecture for a Learning by Demonstration System.

Adapted from Mehta et al. (2009).

are stored in a behavior base that can be used by the planner to achieve goals while playing the game. This results in a system that requires less expert programmer input to develop a game AI because it may be trained to carry out goals and behavior (Mehta et al. 2009).

The system described by Weber and Ontañón (2010) analyzes StarCraft replays to determine the goals being pursued by the player with each action. Using an expert-defined ontology of goals, the system learns which sequences of actions lead to goals being achieved, and in which situations these actions occurred. Thus, it can automatically annotate replays with the goals being undertaken at each point, and convert this knowledge into a case base that is usable in a case-based planning system. The case-based planning system produced was able to play games of StarCraft by retrieving and adapting relevant cases, but was unable to beat the in-built scripted StarCraft AI. Weber and Ontañón (2010) suggest that the system's capability could be improved using more domain knowledge for comparing state features and identifying goals, which would make it more specific to StarCraft but less generally applicable.

An alternative to analyzing traces is to gather the cases in real time as the game is being played and the correct behavior is being demonstrated — known as online learning. This method has been used to train particular desired behaviors in robots learning robot soccer, so that humans could guide the learning process and apply more training if necessary (Grollman and Jenkins 2007). The training of particular desired behaviors in this way meant that fewer training examples could be covered, so while the robot could learn individual behaviors quickly, it required being set into explicit states for each behavior (Grollman and Jenkins 2007). To the authors' knowledge, such an approach has not been attempted in RTS games.

Open Research Areas

As well as the areas covered above, most of which are actively being researched, there are some areas that are applicable to RTS AI but seem to have been given little attention. The first of these areas is found by examining the use of game AI in industry and how it differs from academic AI. The next area — multi-scale AI — has had a few contributions that have yet to be thoroughly examined, while the third — cooperation — is all but absent from the literature. Each of these three areas raises problems that are challenging for AI agents, and yet almost trivial for a

human player. The final section notes the inconsistency in evaluation methods between various papers in the field and calls for a standardized evaluation method to be put into practice.

Game AI in Industry

Despite the active research in the RTS AI field, there seems to be a large divide between the academic research, which uses new, complex AI techniques, and the games industry, which usually uses older and much simpler approaches. By examining the differences in academic and industry use of AI, we see new opportunities for research that may benefit both groups.

Many papers reason that RTS AI research will be useful for new RTS game development by reducing the work involved in creating AI opponents, or by allowing game developers to create better AI opponents (Baekkelund 2006; Dill 2006; Mehta et al. 2009; Ontañón 2012; Ponsen et al. 2005; Tozour 2002; Woodcock 2002). For example, the RTS game DEFCON was given enhanced, learning AI through collaboration with the Imperial College of London (discussed earlier) (Baumgarten, Colton, and Morris 2009). Similarly, Kohan II: Kings of War was produced with flexible AI through a dynamic goal-selection mechanism based on complex priority calculations (discussed earlier) (Dill 2006). More recently, the currently in development RTS game Planetary Annihilation[19] is using flow fields for effective unit pathfinding with large numbers of units, and neural networks for controlling squads of units.[20]

In practice, however, there is very low rate of industry adoption of academic game AI research. It is typical for industry game producers to specify and encode manually the exact behavior of their agents instead of using learning or reasoning techniques (Mehta et al. 2009; Tozour 2002; Woodcock 2002). Older techniques such as scripting, finite state machines, decision trees, and rule-based systems are still the most commonly used (Ontañón 2012; Tozour 2002; Woodcock 2002)[20] — for example, the built-in AI of StarCraft uses a static script that chooses randomly among a small set of predetermined behaviors (Huang 2011). These techniques result in game AI that often has predictable, inflexible behavior, is subject to repeatable exploitation by humans, and doesn't learn or adapt to unforeseen situations or events (Dill 2006; Huang 2011; Ontañón 2012; Woodcock 2002).

There are two main reasons for this lack of adoption of academic AI techniques. Firstly, there is a notable difference in goals between academe and industry. Most academic work focuses on trying to create rational, optimal agents that reason, learn, and react, while the industry aims to create challenging but defeatable opponents that are fun to play against, usually through entirely predefined behavior (Baumgarten, Colton, and Morris 2009; Davis 1999; Lidén 2004; Ontañón 2012; Tozour 2002). The two aims are linked, as players find a game more fun when it is reasonably challenging (Hagelbäck and Johansson 2009),[21] but this difference in goals results in very different behavior from the agents. An agent aiming to play an optimal strategy — especially if it is the same optimal strategy every game — is unlikely to make a desirable RTS opponent, because humans enjoy finding and taking advantage of opportunities and opponent mistakes.[22] An optimal agent is also trying to win at all costs, while the industry really wants game AI that is aiming to lose the game, but in a more humanlike way (Davis 1999). [22] Making AI that acts more humanlike and intelligent — even just in specific circumstances through scripted behaviors — is important in the industry as it is expected to make a game more fun and interesting for the players (Lidén 2004; Scott 2002; Woodcock 2002).

The second major reason for the lack of adoption is that there is little demand from the games industry for new AI techniques. Industry game developers do not view their current techniques as an obstacle to making game AI that is challenging and fun to play against, and note that it is difficult to evaluate the potential of new, untested techniques (Woodcock 2002).[20, 22] Industry RTS games often allow AI opponents to cheat in order to make them more challenging, or emphasize playing against human opponents instead of AI (Davis 1999; Laird and van Lent 2001; Synnaeve and Bessière 2011a). Additionally, game development projects are usually under severe time and resource constraints, so trying new AI techniques is both costly and risky (Buro 2004; Tozour 2002).[20] In contrast, the existing techniques are seen as predictable, reliable, and easy to test and debug (Dill 2006; Baekkelund 2006; Tozour 2002; Woodcock 2002).[22] Academic AI techniques are also seen as difficult to customize, tune, or tweak in order to perform important custom scripted tasks, which scripted AI is already naturally suited to doing.[20, 22]

Some new avenues of research come to light considering the use of game AI in industry. Most importantly, creating AI that is more humanlike, which may also make it more fun to play against. This task could be approached by making an RTS AI that is capable of more difficult human interactions. Compared to AI, human players are good at working together with allies, using surprises, deception, distractions and coordinated attacks, planning effective strategies, and changing strategies to become less predictable (Scott 2002). Players that are able to do at least some of these things appear to be intelligent and are more fun for human players to play against (Scott 2002). In addition, being predictable and exploitable in the same fashion over multiple games means that human players do not get to find and exploit new mistakes, removing a source of enjoyment from the game. AI can even make mistakes and still appear intelligent as long as the mistake appears plausible in

the context of the game — the sort of mistakes that a human would make (Lidén 2004).

An alternative way to create AI that is more humanlike is to replicate human play styles and skills. Enabling an AI to replicate particular strategies — for example a heavily defensive turtle strategy or heavily offensive rush strategy — would give the AI more personality and allow players to practice against particular strategies.[22] This concept has been used in industry AI before (Dill 2006) but may be difficult to integrate into more complex AI techniques. A system capable of learning from a human player — using a technique such as learning from demonstration (see the section on this topic), likely using offline optimization — could allow all or part of the AI to be trained instead of programmed (Floyd and Esfandiari 2010; Mehta et al. 2009). Such a system could potentially copy human skills — like unit micromanagement or building placement — in order to keep up with changes in how humans play a game over time, which makes it an area of particular interest to the industry.[22]

Evaluating whether an RTS AI is humanlike is potentially an issue. For FPS games, there is an AI competition, BotPrize,[20] for creating the most humanlike bots (AI players), where the bots are judged on whether they appear to be a human playing the game — a form of Turing test.[24] This test has finally been passed in 2012, with two bots judged more likely to be humans than bots for the first time. Appearing humanlike in an RTS would be an even greater challenge than in an FPS, as there are more ways for the player to act and react to every situation, and many actions are much more visible than the very fast-paced transient actions of an FPS. However, being humanlike is not currently a focus of any StarCraft AI research, to the authors' knowledge, although it has been explored to a very small extent in the context of some other RTS games. It is also not a category in any of the current StarCraft AI competitions. The reason for this could be the increased difficulty of creating a human level agent for RTS games compared with FPS games, however, it may simply be due to an absence of goals in this area of game AI research. A Turing Test similar to BotPrize could be designed for StarCraft bots by making humans play in matches and then decide whether their opponent was a human or a bot. It could be implemented fairly easily on a competitive ladder like ICCup by simply allowing a human to join a match and asking them to judge the humanness of their opponent during the match. Alternatively, the replay facility in StarCraft could be used to record matches between bots and humans of different skill levels, and other humans could be given the replays to judge the humanness of each player. Due to the popularity of StarCraft, expert participants and judges should be relatively easy to find.

A secondary avenue of research is in creating RTS AI that is more accessible or useful outside of academe. This can partially be addressed by simply considering and reporting how often the AI can be relied upon to behave as expected, how performant the system is, and how easily the system can be tested and debugged. However, explicit research into these areas could yield improvements that would benefit both academe and industry. More work could also be done to investigate how to make complex RTS AI systems easier to tweak and customize, to produce specific behavior while still retaining learning or reasoning capabilities. Industry feedback indicates it is not worthwhile to adapt individual academic AI techniques in order to apply them to individual games, but it may become worthwhile if techniques could be reused for multiple games in a reliable fashion. A generalized RTS AI middleware could allow greater industry adoption — games could be more easily linked to the middleware and then tested with multiple academic techniques — as well as a wider evaluation of academic techniques over multiple games. Research would be required in order to find effective abstractions for such a complex and varied genre of games, and to show the viability of this approach.

Multiscale AI

Due to the complexity of RTS games, current bots require multiple abstractions and reasoning mechanisms working in concert in order to play effectively (Churchill and Buro 2012; Weber et al. 2010; Weber, Mateas, and Jhala 2011a). In particular, most bots have separate ways of handling tactical and strategic level decision making, as well as separately managing resources, construction, and reconnaissance. Each of these modules faces an aspect of an interrelated problem, where actions taken will have long-term strategic trade-offs affecting the whole game, so they cannot simply divide the problem into isolated or hierarchical problems. A straightforward hierarchy of command — like in a real-world military — is difficult in an RTS because the decisions of the top-level commander will depend on, and affect, multiple subproblems, requiring an understanding of each one as well as how they interact. For example, throughout the game, resources could be spent on improving the resource generation, training units for an army, or constructing new base infrastructure, with each option controlled by a different module that cannot assess the others' situations. Notably, humans seem to be able to deal with these problems very well through a combination of on- and offline, reactive, deliberative, and predictive reasoning.

Weber et al. (2010) define the term *multiscale AI problems* to refer to these challenges, characterized by concurrent and coordinated goal pursuit across multiple abstractions. They go on to describe several different approaches they are using to integrate parts of their bot. First is a working memory or shared black-

board concept for indirect communication between their modules, where each module publishes its current beliefs for the others to read. Next, they allow for goals and plans generated by their planning and reasoning modules to be inserted into their central reactive planning system, to be pursued in parallel with current goals and plans. Finally, they suggest a method for altered behavior activation, so that modules can modify the preconditions for defined behaviors, allowing them to activate and deactivate behaviors based on the situation.

A simpler approach may be effective for at least some parts of an RTS bot. Synnaeve and Bessière (2011b) use a higher-level tactical command, such as scout, hold position, flock, or fight, as one of the inputs to their micromanagement controller. Similarly, Churchill and Buro (2012) use a hierarchical structure for unit control, with an overall game commander — the module that knows about the high-level game state and makes strategic decisions — giving commands to a macro commander and a combat commander, each of which give commands to their subcommanders. Commanders further down the hierarchy are increasingly focused on a particular task, but have less information about the overall game state, so therefore must rely on their parents to make them act appropriately in the bigger picture. This is relatively effective because the control of units is more hierarchically arranged than other aspects of an RTS. Such a system allows the low-level controllers to incorporate information from their parent in the hierarchy, but they are unable to react and coordinate with other low-level controllers directly in order to perform cooperative actions (Synnaeve and Bessière 2011b). Most papers on StarCraft AI skirt this issue by focusing on one aspect of the AI only, as can be seen in how this review paper is divided into tactical and strategic decision making sections.

Cooperation

Cooperation is an essential ability in many situations, but RTS games present a particular complex environment in which the rules and overall goal are fixed, and there is a limited ability to communicate with your cooperative partner(s). It would also be very helpful in commercial games, as good cooperative players could be used for coaching or team games. In team games humans often team up to help each other with coordinated actions throughout the game, like attacking and defending, even without actively communicating. Conversely AI players in most RTS games (including StarCraft) will act seemingly independently of their teammates. A possible beginning direction for this research could be to examine some techniques developed for opponent modeling and reuse them for modeling an ally, thus giving insight into how the player should act to coordinate with the ally. Alternatively, approaches to teamwork and coordination used in other domains, such as RoboCup (Kitano et al. 1998) may be appropriate to be adapted or extended for use in the RTS domain.

Despite collaboration being highlighted as a challenging AI research problem in Buro (2003), to the authors' knowledge just one research publication focusing on collaborative behavior exists in the domain of StarCraft (and RTS games in general). Magnusson and Balsasubramaniyan (2012) modified an existing StarCraft bot to allow both communication of the bot's intentions and in-game human control of the bot's behavior. It was tested in a small experiment in which a player is allied with the bot, with or without the communication and control elements, against two other bots. The players rated the communicating bots as more fun to play with than the noncommunicating bots, and more experienced players preferred to be able to control the bot while novice players preferred a noncontrollable bot. Much more research is required to investigate collaboration between humans and bots, as well as collaboration between bots only.

Standardized Evaluation

Despite games being a domain that is inherently suited to evaluating the effectiveness of the players and measuring performance, it is difficult to make fair comparisons between the results of most literature in the StarCraft AI field.

Almost every paper has a different method for evaluating its results, and many of these experiments are of poor quality. Evaluation is further complicated by the diversity of applications, as many of the systems developed are not suited to playing entire games of StarCraft, but are suited to a specific subproblem. Such a research community, made up of isolated studies that are not mutually comparable, was recognized as problematic by Aha and Molineaux (2004). Their Testbed for Integrating and Evaluating Learning Techniques (TIELT), which aimed to standardize the learning environment for evaluation, attempted to address the problem but unfortunately never became very widely used.

Partial systems — those that are unable to play a full game of StarCraft — are often evaluated using a custom metric, which makes comparison between such systems nearly impossible. A potential solution for this would be to select a common set of parts that could plug in to partial systems and allow them to function as a complete system for testing. This may be possible by compartmentalizing parts of an open-source AI used in a StarCraft AI competition, such as UAlbertaBot (Churchill and Buro 2012), which is designed to be modular, or using an add-on library such as the BWAPI Standard Add-on Library (BWSAL).[25] Alternatively, a set of common tests could be made for partial systems to be run against. Such tests could examine common subproblems of an AI system, such as tactical decision making, planning,

and plan recognition, as separate suites of tests. Even without these tests in place, new systems should at least be evaluated against representative related systems in order to show that they represent a nontrivial improvement.

Results published about complete systems are similarly difficult to compare against one another due to their varied methods of evaluation. Some of the only comparable results come from systems demonstrated against the inbuilt StarCraft AI, despite the fact that the inbuilt AI is a simple scripted strategy that average human players can easily defeat (Weber, Mateas, and Jhala 2010). Complete systems are more effectively tested in StarCraft AI competitions, but these are run infrequently, making quick evaluation difficult. An alternative method of evaluation is to automatically test the bots against other bots in a ladder tournament, such as in the StarCraft Brood War Ladder for BWAPI Bots.[26] In order to create a consistent benchmark of bot strength, a suite of tests could be formed from the top three bots from each of the AIIDE StarCraft competitions on a selected set of tournament maps. This would provide enough variety to give a general indication of bot strength, and it would allow for results to be compared between papers and over different years. An alternative to testing bots against other bots is testing them in matches against humans, such as how Weber, Mateas, and Jhala (2010) tested their bot in the ICCup.

Finally, it may be useful to have a standard evaluation method for goals other than finding the AI best at winning the game. For example, the game industry would be more interested in determining the AI that is most fun to play against, or the most humanlike. A possible evaluation for these alternate objectives was discussed earlier.

Conclusion

This article has reviewed the literature on artificial intelligence for real-time strategy games focusing on StarCraft. It found significant research focus on tactical decision making, strategic decision making, plan recognition, and strategy learning. Three main areas were identified where future research could have a large positive impact. First, creating RTS AI that is more humanlike would be an interesting challenge and may help to bridge the gap between academe and industry. The other two research areas discussed were noted to be lacking in research contributions, despite being highly appropriate for real-time strategy game research: multiscale AI, and cooperation. Finally, the article finished with a call for increased rigor and ideally standardization of evaluation methods, so that different techniques can be compared on even ground. Overall the RTS AI field is small but very active, with the StarCraft agents showing continual improvement each year, as well as gradually becoming more based upon machine learning, learning from demonstration, and reasoning, instead of using scripted or fixed behaviors.

Notes

1. Blizzard Entertainment: StarCraft: blizzard.com/games/sc/.

2. Wargus: wargus.sourceforge.net.

3. Open RTS: skatgame.net/mburo/orts.

4. Brood War API: code.google.com/p/bwapi.

5. AIIDE StarCraft AI Competition: www.starcraftaicompetition.com.

6. CIG StarCraft AI Competition: ls11-www.cs.uni-dortmund.de/rts-competition/.

7. Mad Doc Software. Website no longer available.

8. SparCraft: code.google.com/p/sparcraft/.

9. Blizzard Entertainment: Warcraft III: blizzard.com/games/war3/.

10. TimeGate Studios: Kohan II Kings of War: www.timegate.com/games/kohan-2-kings-of-war.

11. Spring RTS: springrts.com.

12. International Cyber Cup: www.iccup.com.

13. See A. J. Champandard, This Year [2010] in Game AI: Analysis, Trends from 2010 and Predictions for 2011. aigamedev.com/open/editorial/2010-retrospective.

14. Blizzard Entertainment: StarCraft II: blizzard.com/games/sc2/.

15. Evolution Chamber: code.google.com/p/evolution-chamber/.

16. See A. Turner, 2012, Soar-SC: A Platform for AI Research in StarCraft: Brood War github.com/bluechill/Soar-SC/tree/master/Soar-SC-Papers.

17. Introversion Software: DEFCON: www.introversion.co.uk/defcon.

18. RoboCup: www.robocup.org.

19. Uber Entertainment: Planetary Annihilation: www.uberent.com/pa.

20. Personal communication with M. Robbins, 2013. Robbins is a software engineer at Uber Entertainment, formerly game-play engineer at Gas Powered Games.

21. Also see L. Dicken's 2011 blog, altdevblogaday.com/2011/05/12/a-difficult-subject/.

22. Personal communication with B. Schwab, 2013. Schwab is a senior AI/game-play engineer at Blizzard Entertainment.

23. BotPrize: botprize.org.

24. See L. Dicken's 2011 blog, A Turing Test for Bots. altdevblogaday.com/2011/09/09/a-turing-test-for-bots/.

25. BWAPI Standard Add-on Library: code.google.com/p/bwsal.

26. StarCraft Brood War Ladder for BWAPI Bots: bots-stats.krasi0.com.

References

Aamodt, A., and Plaza, E. 1994. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications* 7(1): 39–59.

Aha, D.; Molineaux, M.; and Ponsen, M. 2005. Learning to Win: Case-Based Plan Selection in a Real-Time Strategy Game. In *Case-Based Reasoning: Research and Development,*

volume 3620, Lecture Notes in Computer Science, ed. H. Muñoz-Ávila and F. Ricci, 5–20. Berlin: Springer.

Aha, D. W., and Molineaux, M. 2004. Integrating Learning in Interactive Gaming Simulators. In Challenges in Game AI: Papers from the AAAI Workshop. AAAI Technical Report WS-04-04. Palo Alto, CA: AAAI Press.

Baekkelund, C. 2006. Academic AI Research and Relations with the Games Industry. In *AI Game Programming Wisdom,* volume 3, ed. S. Rabin, 77–88. Boston, MA: Charles River Media.

Bakkes, S.; Spronck, P.; and van den Herik, J. 2011. A CBR-Inspired Approach to Rapid and Reliable Adaption of Video Game AI. Paper presented at the Case-Based Reasoning for Computer Games Workshop at the International Conference on Case-Based Reasoning (ICCBR), 17–26. Greenwich, London, 12–15 September.

Balla, R., and Fern, A. 2009. UCT for Tactical Assault Planning in Real-Time Strategy Games. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence* (IJCAI), 40–45. Palo Alto, CA: AAAI Press.

Baumgarten, R.; Colton, S.; and Morris, M. 2009. Combining AI Methods for Learning Bots in a Real-Time Strategy Game. *International Journal of Computer Games Technology* 2009: Article Number 4.

Buckland, M. 2005. *Programming Game AI by Example*. Plano, TX: Wordware Publishing, Inc.

Buro, M., ed. 2012. Artificial Intelligence in Adversarial Real-Time Games: Papers from the 2012 AIIDE Workshop, AAAI Technical Report WS-12-15. Palo Alto, CA: AAAI Press.

Buro, M. 2004. Call for AI Research in RTS Games. In Challenges in Game AI: Papers from the AAAI Workshop, 139–142. AAAI Technical Report WS-04-04. Palo Alto, CA: AAAI Press.

Buro, M. 2003. Real-Time Strategy Games: A New AI Research Challenge. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence,* 1534–1535. San Francisco: Morgan Kaufmann, Inc.

Buro, M., and Churchill, D. 2012. Real-Time Strategy Game Competitions. *AI Magazine* 33(3): 106–108.

Buro, M., and Furtak, T. M. 2004. RTS Games and Real-Time AI Research. Paper presented at the 13th Behavior Representation in Modeling and Simulation Conference, Arlington, VA, USA, 17–20 May.

Cadena, P., and Garrido, L. 2011. Fuzzy Case-Based Reasoning for Managing Strategic and Tactical Reasoning in StarCraft. In *Advances in Artificial Intelligence,* volume 7094, Lecture Notes in Computer Science, ed. I. Batyrshin and G. Sidorov, 113–124. Berlin: Springer.

Chan, H.; Fern, A.; Ray, S.; Wilson, N.; and Ventura, C. 2007. Online Planning for Resource Production in Real-Time Strategy Games. In *Proceedings of the 17th International Conference on Automated Planning and Scheduling* (ICAPS), 65–72. Menlo Park, CA: AAAI Press.

Cheng, D., and Thawonmas, R. 2004. Case-Based Plan Recognition for Real-Time Strategy Games. In *Proceedings of the 5th Annual European GAME-ON Conferenc*e, 36–40. Reading, UK: University of Wolverhampton Press.

Chung, M.; Buro, M.; and Schaeffer, J. 2005. Monte Carlo Planning in RTS Games. In Kendall, G., and Lucas, S., eds., *Proceedings of the 2005 IEEE Conference on Computational Intelligence and Games,* 117–124. Piscataway, NJ: Institute for Electrical and Electronics Engineers.

Churchill, D., and Buro, M. 2011. Build Order Optimization in StarCraft. In *Proceedings of the Seventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment,* 14–19. Palo Alto, CA: AAAI Press.

Churchill, D., and Buro, M. 2012. Incorporating Search Algorithms into RTS Game Agents. In Artificial Intelligence in Adversarial Real-Time Games: Papers from the 2012 AIIDE Workshop, AAAI Technical Report WS-12-15, 2–7. Palo Alto, CA: AAAI Press.

Churchill, D.; Saffidine, A.; and Buro, M. 2012. Fast Heuristic Search for RTS Game Combat Scenarios. In *Proceedings of the Eighth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment,* 112–117. Palo Alto, CA: AAAI Press

Davis, I. L. 1999. Strategies for Strategy Game AI. In Artificial Intelligence and Computer Games: Papers from the AAAI Spring Symposium, 24–27, AAAI Technical Report SS-99-02. Menlo Park, CA: AAAI Press.

Dereszynski, E.; Hostetler, J.; Fern, A.; Dietterich, T.; Hoang, T.; and Udarbe, M. 2011. Learning Probabilistic Behavior Models in Real-Time Strategy Games. In *Proceedings of the Seventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment,* 20–25. Palo Alto, CA: AAAI Press.

Dill, K. 2006. Prioritizing Actions in a Goal-Based RTS AI. In *AI Game Programming Wisdom,* volume 3, ed. S. Rabin, 321–330. Boston, MA: Charles River Media.

Floyd, M., and Esfandiari, B. 2009. Comparison of Classifiers for Use in a Learning by Demonstration System for a Situated Agent. Paper Presented at the Case-Based Reasoning for Computer Games at the 8th International Conference on Case-Based Reasoning, Seattle, WA, USA, 20–23 July.

Floyd, M., and Esfandiari, B. 2010. Toward a Domain Independent Case-Based Reasoning Approach for Imitation: Three Case Studies in Gaming. Paper Presented at the Case-Based Reasoning for Computer Games Workshop held at the 18th International Conference on Case-Based Reasoning, Alessandria, Italy, 19–22 July.

Floyd, M. W., and Esfandiari, B. 2011a. A Case-Based Reasoning Framework for Developing Agents Using Learning by Observation. In *Proceedings of the 2011 IEEE International Conference on Tools with Artificial Intelligence,* 531–538. Piscataway, NJ: Institute for Electrical and Electronics Engineers.

Floyd, M., and Esfandiari, B. 2011b. Learning State-Based Behaviour Using Temporally Related Cases. Presented at the Nineteenth UK Workshop on Case-Based Reasoning, Cambridge, UK, 9 December.

Gabriel, I.; Negru, V.; and Zaharie, D. 2012. Neuroevolution Based MultiAgent System for Micromanagement in Real-Time Strategy Games. In *Proceedings of the Fifth Balkan Conference in Informatics,* 32–39. New York: Association for Computing Machinery.

Grollman, D., and Jenkins, O. 2007. Learning Robot Soccer Skills From Demonstration. In *Proceedings of the 2007 IEEE International Conference on Development and Learning,* 276–281. Piscataway, NJ: Institute for Electrical and Electronics Engineers.

Hagelbäck, J., and Johansson, S. J. 2008. The Rise of Potential Fields in Real Time Strategy Bots. In *Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference,* 42–47. Palo Alto, CA: AAAI Press.

Hagelbäck, J., and Johansson, S. 2009. Measuring Player Experience on Runtime Dynamic Difficulty Scaling in an

RTS Game. In *Proceedings of the 2009 IEEE Conference on Computational Intelligence and Games,* 46–52. Piscataway, NJ: Institute for Electrical and Electronics Engineers.

Hostetler, J.; Dereszynski, E.; Dietterich, T.; and Fern, A. 2012. Inferring Strategies from Limited Reconnaissance in Real-Time Strategy Games. Paper presented at the Conference on Unceertainty in Artificial Intelligence, Avalon, Catalina Island, CA, 15–17 August.

Hsieh, J., and Sun, C. 2008. Building a Player Strategy Model by Analyzing Replays of Real-Time Strategy Games. In *Proceedings of the 2008 IEEE International Joint Conference on Neural Networks,* 3106–3111. Piscataway, NJ: Institute for Electrical and Electronics Engineers.

Huang, H. 2011. Skynet Meets the Swarm: How the Berkeley Overmind Won the 2010 StarCraft AI Competition. *Ars Technica* 18 January 2011. (arstechnica.com/gaming/news/2011/01/skynet-meets-the-swarm-how-the-berkeley-overmind-won-the-2010-starcraft-ai-competition.ars).

Jaidee, U.; Muñoz-Avila, H.; and Aha, D. 2011. Integrated Learning for Goal-Driven Autonomy. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence,* 2450–2455. Palo Alto, CA: AAAI Press.

Judah, K.; Roy, S.; Fern, A.; and Dietterich, T. G. 2010. Reinforcement Learning via Practice and Critique Advice. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*. Palo Alto, CA: AAAI Press.

Kabanza, F.; Bellefeuille, P.; Bisson, F.; Benaskeur, A.; and Irandoust, H. 2010. Opponent Behaviour Recognition for Real-Time Strategy Games. In Plan, Activity, and Intent Recognition: Papers from the AAAI Workshop. Technical Report WS-10-15. Palo Alto, CA: AAAI Press.

Kitano, H.; Tambe, M.; Stone, P.; Veloso, M.; Coradeschi, S.; Osawa, E.; Matsubara, H.; Noda, I.; and Asada, M. 1998. The Robocup Synthetic Agent Challenge 97. In *RoboCup-97: Robot Soccer World Cup I,* volume 1395, Lecture Notes in Computer Science, ed. H. Kitano, 62–73. Berlin: Springer.

Laagland, J. 2008. A HTN Planner for a Real-Time Strategy Game. Unpublished manuscript. (hmi.ewi.utwente.nl/verslagen/capita-selecta/CS-Laagland-Jasper.pdf).

Laird, J., and van Lent, M. 2001. Human-Level AI'S Killer Application: Interactive Computer Games. *AI Magazine* 22(2): 15–26.

Lidén, L. 2004. Artificial Stupidity: The Art of Intentional Mistakes. In *AI Game Programming Wisdom,* volume 2, ed. S. Ragin, 41–48. Hingham, MA: Charles River Media.

Magnusson, M. M., and Balsasubramaniyan, S. K. 2012. A Communicating and Controllable Teammate Bot for RTS Games. Master's thesis, School of Computing, Blekinge Institute of Technology, Blekinge Sweden.

Manslow, J. 2004. Using Reinforcement Learning to Solve AI Control Problems. In *AI Game Programming Wisdom*, volume 2, ed. Rabin, S. Hingham, MA: Charles River Media. 591–601.

Marthi, B.; Russell, S.; Latham, D.; and Guestrin, C. 2005. Concurrent Hierarchical Reinforcement Learning. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence,* 779–785. San Francisco: Morgan Kaufmann, Inc.

Mateas, M., and Stern, A. 2002. A Behavior Language for Story-Based Believable Agents. *IEEE Intelligent Systems* 17(4): 39–47.

Mehta, M.; Ontañón, S.; Amundsen, T.; and Ram, A. 2009. Authoring Behaviors for Games Using Learning from Demonstration. Paper Presented at the Case-Based Reasoning for Computer Games at the 8th International Conference on Case-Based Reasoning, Seattle, WA, USA, 20–23 July.

Mishra, K.; Ontañón, S.; and Ram, A. 2008. Situation Assessment for Plan Retrieval in Real-Time Strategy Games. In *Advances in Case-Based Reasoning,* volume 5239, Lecture Notes in Computer Science, ed. K.-D. Althoff, R. Bergmann, M. Minor, and A. Hanft, 355–369. Berlin: Springer.

Molineaux, M.; Aha, D.; and Moore, P. 2008. Learning Continuous Action Models in a Real-Time Strategy Environment. In *Proceedings of the Twenty-First International Florida Artificial Intelligence Research Society (FLAIRS) Conference,* 257–262. Palo Alto, CA: AAAI Press.

Molineaux, M.; Klenk, M.; and Aha, D. 2010. Goal-Driven Autonomy in a Navy Strategy Simulation. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence.* Palo Alto, CA: AAAI Press.

Muñoz-Avila, H., and Aha, D. 2004. On the Role of Explanation for Hierarchical Case-Based Planning in Real-Time Strategy Games. In *Advances in Case-Based Reasoning, 7th European Conference,* ECCBR 2004. Lecture Notes in Computer Science. Berlin: Springer.

Nejati, N.; Langley, P.; and Konik, T. 2006. Learning Hierarchical Task Networks by Observation. In *Proceedings of the 23rd International Conference on Machine Learning,* 665–672. New York: Association for Computing Machinery.

Ontañón, S. 2012. Case Acquisition Strategies for Case-Based Reasoning in Real-Time Strategy Games. In *Proceedings of the Twenty-Fifth International Florida Artificial Intelligence Research Society Conference* (FLAIRS). Palo Alto, CA: AAAI Press.

Ontañón, S.; Mishra, K.; Sugandh, N.; and Ram, A. 2007. Case-Based Planning and Execution for Real-Time Strategy Games. In *Case-Based Reasoning: Research and Development,* volume 4626, Lecture Notes in Computer Science, ed. R. Weber and M. Richter, 164–178. Berlin: Springer.

Ontañón, S.; Montana, J.; and Gonzalez, A. 2011. Towards a Unified Framework for Learning from Observation. Paper presented at the Workshop on Agents Learning Interactively from Human Teachers at the Twenty-Second International Joint Conference on Artificial Intelligence, Barcelona, Spain 16–22 July.

Ontañón, S.; Synnaeve, G.; Uriarte, A.; Richoux, F.; Churchill, D.; and Preuss, M. In press. A Survey of Real-Time Strategy Game AI Research and Competition in StarCraft. *Transactions of Computational Intelligence and AI in Games* 5(4): 1–19.

Orkin, J. 2004. Applying Goal-Oriented Action Planning to Games. In *AI Game Programming Wisdom,* volume 2, ed. S. Rabin, 217–227. Hingham, MA: Charles River Media.

Palma, R.; Sánchez-Ruiz, A.; Gómez-Martín, M.; Gómez-Martín, P.; and González-Calero, P. 2011. Combining Expert Knowledge and Learning from Demonstration in Real-Time Strategy Games. In *Case-Based Reasoning Research and Development,* volume 6880, Lecture Notes in Computer Science, ed. A. Ram and N. Wiratunga, 181–195. Berlin: Springer.

Perkins, L. 2010. Terrain Analysis in Real-Time Strategy Games: an Integrated Approach to Choke Point Detection and Region Decomposition. In *Proceedings of the Sixth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment,* 168–173. Palo Alto, CA: AAAI Press.

Ponsen, M.; Muñoz-Avila, H.; Spronck, P.; and Aha, D. 2005. Automatically Acquiring Domain Knowledge for Adaptive Game AI Using Evolutionary Learning. In *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference,* 1535–1540. Palo Alto, CA: AAAI Press.

Ponsen, M.; Muñoz-Avila, H.; Spronck, P.; and Aha, D. 2006. Automatically Generating Game Tactics Through Evolutionary Learning. *AI Magazine* 27(3): 75–84.

Sailer, F.; Buro, M.; and Lanctot, M. 2007. Adversarial Planning Through Strategy Simulation. In *Proceedings of the IEEE Conference on Computational Intelligence and Games,* 80–87. Piscataway, NJ: Institute for Electrical and Electronics Engineers.

Sánchez-Pelegrín, R.; Gómez-Martín, M.; and Díaz-Agudo, B. 2005. A CBR Module for a Strategy Videogame. Paper presented at the ICCBR05 Workshop on Computer Gaming and Simulation Environments at the ICCBR, Chicago, IL, 23–26 August.

Schaeffer, J. 2001. A Gamut of Games. *AI Magazine* 22(3): 29–46.

Scott, B. 2002. The Illusion of Intelligence. In *AI Game Programming Wisdom,* volume 1, ed. S. Rabin, 16–20. Hingham, MA: Charles River Media.

Shantia, A.; Begue, E.; and Wiering, M. 2011. Connectionist Reinforcement Learning for Intelligent Unit Micro Management in StarCraft. Paper Presented at the 2011 International Joint Conference on Neural Networks (IJCNN). San Jose, CA USA, 31 July–5 August.

Sharma, M.; Holmes, M.; Santamaria, J.; Irani, A.; Isbell, C.; and Ram, A. 2007. Transfer Learning in Real-Time Strategy Games Using Hybrid CBR/RL. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*. Palo Alto, CA: AAAI Press.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. Cambridge Massachusetts: The MIT Press.

Synnaeve, G., and Bessière, P. 2011a. A Bayesian Model for Plan Recognition in RTS Games Applied to StarCraft. In *Proceedings of the Seventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment,* 79–84. Palo Alto, CA: AAAI Press.

Synnaeve, G., and Bessière, P. 2011b. A Bayesian Model for RTS Units Control Applied to StarCraft. In Proceedings of the 2011 IEEE Conference on Computational Intelligence and Games, 190–196. Piscataway, NJ: Institute for Electrical and Electronics Engineers.

Synnaeve, G., and Bessière, P. 2012. A Dataset for StarCraft AI and an Example of Armies Clustering. In Artificial Intelligence in Adversarial Real-Time Games: Papers from the 2012 AIIDE Workshop, AAAI Technical Report WS-12-15. Palo Alto, CA: AAAI Press.

Szczepanski, T., and Aamodt, A. 2009. Case-Based Reasoning for Improved Micromanagement in Real-Time Strategy Games. Paper Presented at the Case-Based Reasoning for Computer Games at the 8th International Conference on Case-Based Reasoning, Seattle, WA, USA, 20–23 July.

Tozour, P. 2002. The Evolution of Game AI. In *AI Game Programming Wisdom,* volume 1, ed. S. Ragin, 3–15. Hingham, MA: Charles River Media.

Uriarte, A., and Ontañón, S. 2012. Kiting in RTS Games Using Influence Maps. In Artificial Intelligence in Adversarial Real-Time Games: Papers from the AIIDE Workshop. Technical Report WS-12-14. Palo Alto, CA: AAAI Press.

Weber, B., and Mateas, M. 2009. A Data Mining Approach to Strategy Prediction. In *Proceedings of the 2009 IEEE Symposium on Computational Intelligence and Games,* 140–147. Piscataway, NJ: Institute for Electrical and Electronics Engineers.

Weber, B.; Mateas, M.; and Jhala, A. 2010. Applying Goal-Driven Autonomy to StarCraft. In *Proceedings of the Sixth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 101–106. Palo Alto, CA: AAAI Press.

Weber, B.; Mateas, M.; and Jhala, A. 2011a. Building Human-Level AI for Real-Time Strategy Games. In Advances in Cognitive Systems: Papers from the AAAI Fall Symposium. Technical Report FS-11-01, 329–336. Palo Alto, CA: AAAI Press.

Weber, B.; Mateas, M.; and Jhala, A. 2011b. A Particle Model for State Estimation in Real-Time Strategy Games. In *Proceedings of the Seventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment,* 103–108. Palo Alto, CA: AAAI Press.

Weber, B.; Mateas, M.; and Jhala, A. 2012. Learning from Demonstration for Goal-Driven Autonomy. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence,* 1176–1182. Palo Alto, CA: AAAI Press.

Weber, B.; Mawhorter, P.; Mateas, M.; and Jhala, A. 2010. Reactive Planning Idioms for multiScale Game AI. In *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games,* 115–122. Piscataway, NJ: Institute for Electrical and Electronics Engineers.

Weber, B., and Ontañón, S. 2010. Using Automated Replay Annotation for Case-Based Planning in Games. Paper Presented at the Case-Based Reasoning for Computer Games at the 8th International Conference on Case-Based Reasoning, Seattle, WA, USA, 20–23 July.

Wintermute, S.; Xu, J.; and Laird, J. 2007. SORTS: A Human-Level Approach to Real-Time Strategy AI. In *Proceedings of the Eighth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment,* 55–60. Palo Alto, CA: AAAI Press.

Woodcock, S. 2002. Foreword. In *AI Techniques for Game Programming,* ed. M. Buckland. Portland, OR: Premier Press.

Ian Watson is an associate professor of artificial intellligence in the Department of Computer Science at the University of Auckland, New Zealand. With a background in expert systems Watson became interested in case-based reasoning (CBR) to reduce the knowledge engineering bottleneck. Watson has remained active in CBR, focusing on game AI alongside other techniques. Watson also has an interest in the history of computing, having written a popular science book called *The Universal Machine*.

Glen Robertson is a Ph.D. candidate at the University of Auckland, working under the supervision of Ian Watson. Robertson's research interests are in machine learning and artificial intelligence, particularly in unsupervised learning for complex domains with large data sets.

Power to the People: The Role of Humans in Interactive Machine Learning

Saleema Amershi, Maya Cakmak, W. Bradley Knox, Todd Kulesza[1]

■ *Systems that can learn interactively from their end-users are quickly becoming widespread. Until recently, this progress has been fueled mostly by advances in machine learning; however, more and more researchers are realizing the importance of studying users of these systems. In this article we promote this approach and demonstrate how it can result in better user experiences and more effective learning systems. We present a number of case studies that demonstrate how interactivity results in a tight coupling between the system and the user, exemplify ways in which some existing systems fail to account for the user, and explore new ways for learning systems to interact with their users. After giving a glimpse of the progress that has been made thus far, we discuss some of the challenges we face in moving the field forward.*

Machine learning is a powerful tool for transforming data into computational models that can drive user-facing applications. However, potential users of such applications, who are often domain experts for the application, have limited involvement in the process of developing them. The intricacies of applying machine-learning techniques to everyday problems have largely restricted their use to skilled practitioners. In the traditional applied machine-learning workflow, these practitioners collect data, select features to represent the data, preprocess and transform the data, choose a representation and learning algorithm to construct the model, tune parameters of the algorithm, and finally assess the quality of the resulting model. This assessment often leads to further iterations on many of the previous steps. Typically, any end-user involvement in this process is mediated by the practitioners and is limited to providing data, answering domain-related questions, or giving feedback about the learned model. This results in a design process with lengthy and asynchronous iterations and limits the end users' ability to affect the resulting models.

Consider the following case study of machine-learning practitioners working with biochemists to develop a protein taxonomy by clustering low-level protein structures (Caruana et al. 2006). The project lead recounted their experience in an invited talk at the Intelligent User Interfaces's 2013 Workshop on Interactive Machine Learning (Amershi et al. 2013). First, the practitioners would create a clustering of the protein structures. Then, they would meet with the biochemists to discuss the results. The biochemists would critique the results (for example, "these two proteins should / should not be in the same cluster" or "this cluster is too small"), providing new constraints for the next iteration. Following each meeting, the practitioners would carefully adjust the clustering parameters to adhere to the given con-

 ISSN 0738-4602

straints and recompute clusters for the next meeting. Frustrated by the inefficiency of this laborious process, Caruana and colleagues went on to develop learning algorithms that enable interactive exploration of the clustering space and incorporation of new clustering constraints (Cohn, Caruana, and McCallum 2003; Caruana et al. 2006). These algorithms were intended to give people the ability to rapidly iterate and inspect many alternative clusterings within a single sitting.

Their later approach is an example of interactive machine learning, where learning cycles involve more rapid, focused, and incremental model updates than in the traditional machine-learning process. These properties enable everyday users to interactively explore the model space and drive the system toward an intended behavior, reducing the need for supervision by practitioners. Consequently, interactive machine learning can facilitate the democratization of applied machine learning, empowering end users to create machine-learning-based systems for their own needs and purposes. However, enabling effective end-user interaction with interactive machine learning introduces new challenges that require a better understanding of end-user capabilities, behaviors, and needs.

This article promotes the empirical study of the users of interactive machine-learning systems as a method for addressing these challenges. Through a series of case studies, we illustrate the following propositions:

> Rapid, focused, and incremental learning cycles result in a tight coupling between the user and the system, where the two influence one another. As a result it is difficult to decouple their influence on the resulting model and study such systems in isolation.
>
> Explicitly studying user interaction can challenge assumptions of traditional learning systems about users and better inform the design of interactive learning systems.
>
> The ways in which end users interact with learning systems can be expanded to ways in which practitioners do (for example, tuning parameters or defining new constraints); however, novel interaction techniques should be carefully evaluated with potential end users.

While the presented case studies paint a broad picture of recent research in user interaction with interactive machine learning, this article does not exhaustively survey the literature in this space. Rather, these case studies are selected to highlight the role and importance of the user within the interactive machine-learning process, serving as an introduction to the topic and a vehicle for considering this body of research altogether. We conclude this article with a discussion of the current state of the field, identifying opportunities and open challenges for future interactive machine-learning research.

Interactive Machine Learning

The applied machine-learning workflow often involves long and complex iterations. The process starts with data provided by domain experts or specifically collected for the target application. Machine-learning practitioners then work with domain experts to identify features to represent the data. Next, the practitioners experiment with different machine-learning algorithms, iteratively tuning parameters, tweaking features, and sometimes collecting more data to improve target performance metrics. Results are then further examined both by practitioners and domain experts to inform the subsequent iteration. At the end of this long cycle, the model is updated in several ways and can be drastically different from the previous iteration. Furthermore, this iterative exploration of the model space is primarily driven by the machine-learning practitioners, who rely on their understanding of machine-learning techniques to make informed model updates in each iteration.

In contrast, model updates in interactive machine learning are more rapid (the model gets updated immediately in response to user input), focused (only a particular aspect of the model is updated), and incremental (the magnitude of the update is small; the model does not change drastically with a single update). This allows users to interactively examine the impact of their actions and adapt subsequent inputs to obtain desired behaviors. As a result of these rapid interaction cycles, even users with little or no machine-learning expertise can steer machine-learning behaviors through low-cost trial and error or focused experimentation with inputs and outputs. Figure 1 illustrates traditional applied machine learning and interactive machine learning, highlighting their contrasting characteristics.

Perhaps the most familiar examples of interactive machine learning in real-world applications are recommender systems such as Amazon product recommendations, Netflix movie recommendations, and Pandora music recommendations. Users of recommender systems are often asked targeted questions about their preferences for individual items[2] (which they provide by liking or disliking them, for example). These preferences are then promptly incorporated in the underlying learning system for subsequent recommendations. If a recommender system begins recommending undesired items after incorporating new preferences, the user may attempt to redirect the system by correcting it or providing different preference information in the future.

We next present two case studies that exemplify the interactive machine-learning process and demonstrate its potential as an end-user tool.

Interactive Machine Learning for Image Segmentation

Fails and Olsen (2003) were the first to introduce the

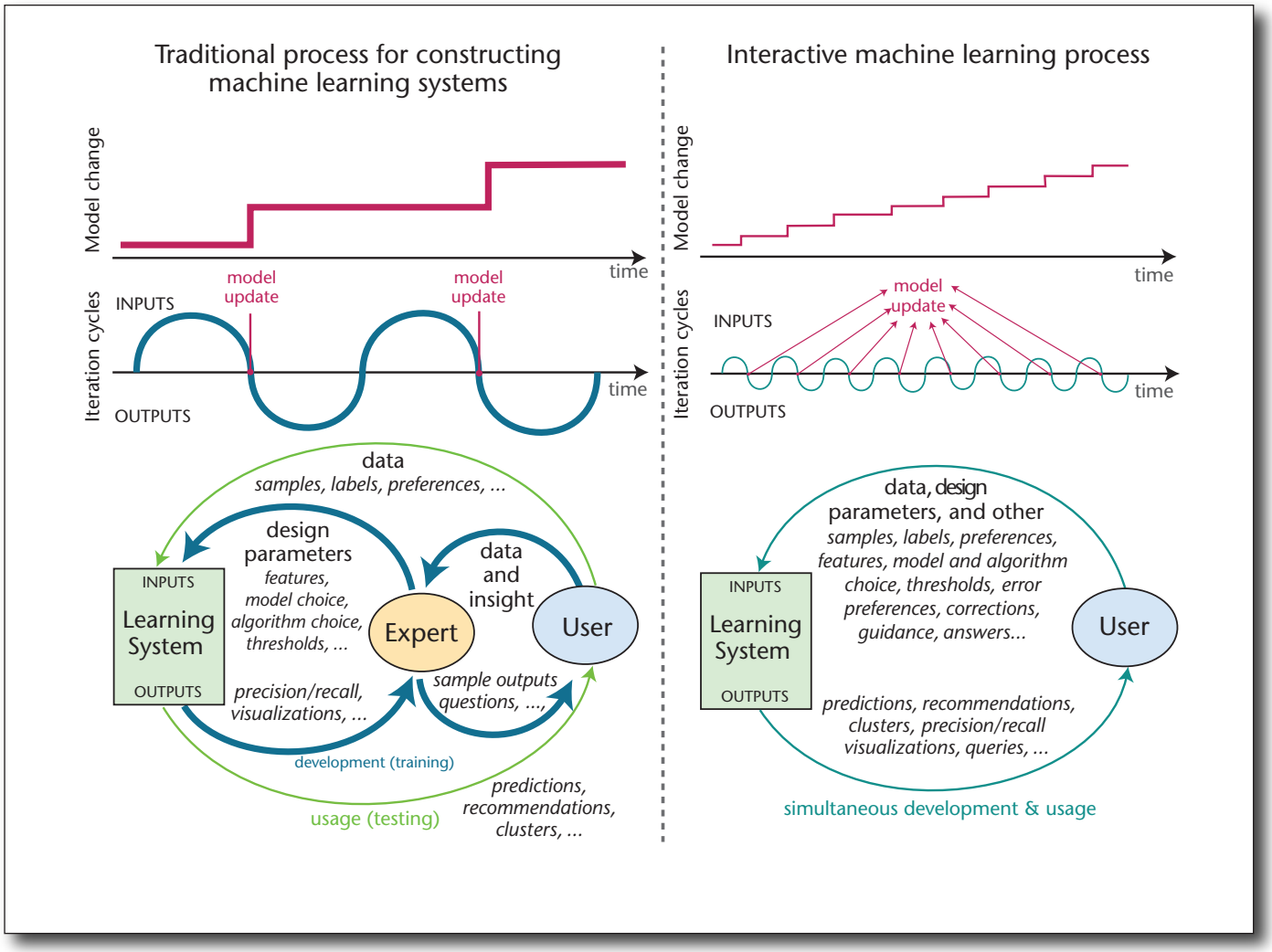


Figure 1. Traditional Applied and Interactive Machine Learning.

In machine learning, people iteratively supply information to a learning system and then observe and interpret the outputs of the system to inform subsequent iterations. In interactive machine learning, these iterations are more focused, frequent, and incremental than traditional machine learning. The tighter interaction between users and learning systems in interactive machine learning necessitates an increased focus on studying the user's involvement in the process.

term *interactive machine learning* in the human-computer interaction community, characterizing it with rapid train-feedback-correct cycles, where users iteratively provide corrective feedback to a learner after viewing its output. They demonstrated this process with their Crayons system, which allowed users with no machine-learning background to train pixel classifiers by iteratively marking pixels as foreground or background through brushstrokes on an image. After each user interaction, the system responded with an updated image segmentation for further review and corrective input.

Evaluations of Crayons through user studies revealed that the immediate output provided by the system allowed users to quickly view and correct misclassifications by adding new training data in the most problematic areas. As illustrated in figure 2, after an initial classification, the user provides Crayons with more data at the edges of the hand where the classifier failed. When asked what they were thinking while interacting with the system, most users stated that they were focused on seeing parts of the image that were classified incorrectly.

Fails and Olsen's work on Crayons demonstrated that users modify their behavior based on a learner's outputs, which is an underlying premise for much of the following research on interactive machine learning.

Interactive Machine Learning for Gesture-Based Music

Another example of an interactive machine-learning system comes from the realm of music composition

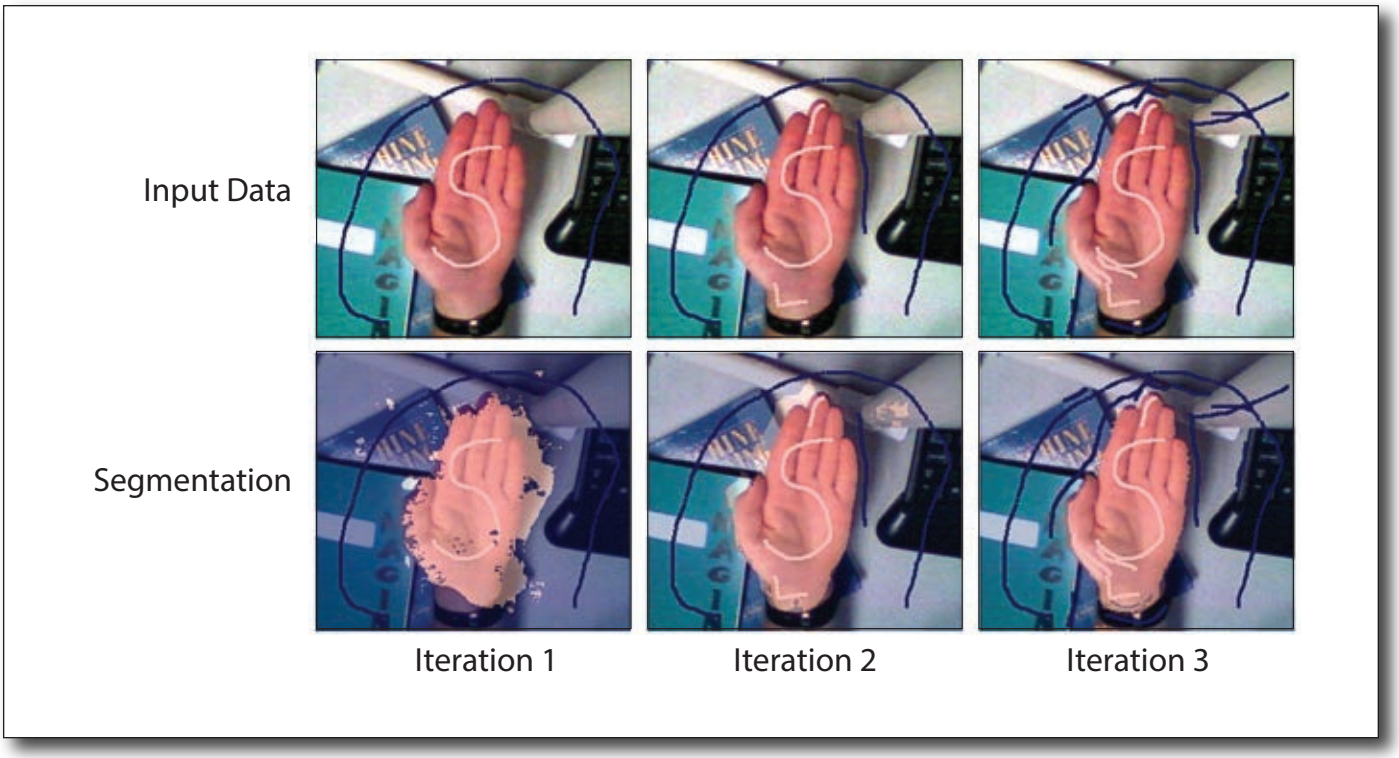


Figure 2. Interactive Training of the Crayons System.

The system takes pixels labeled as background/foreground as input (provided through brush strokes), and gives a fully segmented image as output (obtained through a classifier that labels each pixel as foreground/background). The user's input is focused on areas where the classifier is failing in previous iterations (Fails and Olsen 2003).

and performance. This domain is naturally interactive: musicians are accustomed to receiving immediate feedback when interacting with a musical instrument. Fiebrink, Cook, and Trueman (2011) developed the Wekinator, a machine-learning system for enabling people to interactively create novel gesture-based instruments, such as moving an arm in front of a web camera to produce different sounds based on the arm's position, speed, or rotation. In this system, a neural network receives paired gestures and sounds from the user as input and learns how to interpolate from unobserved gesture positions to a range of sounds. Users evaluate their instruments directly by gesturing and assessing the produced sounds.

While observing students using Wekinator in an interdisciplinary music and computer science course, the authors found that as students trained their respective instruments, the interactive nature of the system also helped train the students. For example, the students learned how to recognize noise in their training samples and provide clearer examples to the learner. In some cases, students even adjusted their goals to match the observed capabilities of the learner. In a follow-up investigation with a professional cellist (Fiebrink, Cook, and Trueman 2011), the cellist identified flaws in her playing technique while trying to train a gesture recognizer. The process revealed that the cellist's bowing articulation was not as precise as she had believed. By observing the outputs of the system in real time, Wekinator users were able to modify their behavior in ways that allowed them to create instruments to their satisfaction.

Summary

These examples illustrate the rapid, focused, and incremental interaction cycles fundamental to interactive machine learning; it is these cycles that facilitate end-user involvement in the machine-learning process. These cycles also result in a tight coupling between the user and the system, making it impossible to study the system in isolation from the user. This necessitates an increased focus on studying how users can effectively influence the machine-learning system and how the learning system can appropriately influence the users. The following section examines how explicitly studying end users can challenge assumptions of traditional machine learning and better inform the development of interactive machine-learning systems. Many of the case studies to follow additionally consider less traditional types of input and output, moving beyond labeled examples and observations of learner predictions.

Figure 3. Users Teaching New Concepts to a Robot by Providing Positive and Negative Examples.

Left: Passive learning: examples are chosen and presented by the user. *Right:* Active learning: particular examples are requested by the learner. Although active learning results in faster convergence, users get frustrated from having to answer the learner's long stream of questions and not having control over the interaction.

Studying User Interaction with Interactive Machine Learning

The increased interaction between users and learning systems in interactive machine learning necessitates an increased understanding of how end-user involvement affects the learning process. In this section, we present case studies illustrating how such an understanding can ultimately lead to better-informed system designs. First, we present case studies demonstrating how people may violate assumptions made by traditional machine learners, resulting in unexpected outcomes and user frustration. Next, we present case studies indicating that people may want to interact with machine-learning systems in richer ways than anticipated, suggesting new input and output capabilities. Finally, we present case studies that experiment with increasing transparency about how machine-learning systems work, finding that such transparency can improve the user experience in some scenarios, as well as the accuracy of resulting models.

Users Are People, Not Oracles

Active learning is a machine-learning paradigm in which the learner chooses the examples from which it will learn (Settles 2010). These examples are selected from a pool of unlabeled samples based on some selection criterion (for example, samples for which the learner has maximum uncertainty). For each selected sample the learner queries an oracle to request a label. This method has had success in accelerating learning (that is, requiring fewer labels to reach a target accuracy) in applications like text classification and object recognition, where oracles are often paid to provide labels over a long period of time. However, Cakmak and colleagues (2010) discovered that when applied to interactive settings, such as a person teaching a task to a robot by example, active learning can cause several problems.

Cakmak's study (figure 3) found that the constant stream of questions from the robot during the interaction was perceived as imbalanced and annoying. The stream of questions also led to a decline in the user's mental model of how the robot learned, causing some participants to "turn their brain off" or "lose track of what they were teaching" (according to their self report) (Cakmak, Choa, and Thomaz 2010). Guillory and Bilmes (2011) reported similar findings for an active movie recommendation system they developed for Netflix. These studies reveal that users are not necessarily willing to be simple oracles (that is, repeatedly telling the computer whether it is right or wrong), breaking a fundamental assumption of active learning. Instead, these systems need to account for human factors such as interruptibility or frustration when employing methods like active learning.

People Tend to Give More Positive Than Negative Feedback to Learners

In reinforcement learning, an agent senses and acts in a task environment and receives numeric reward values after each action. With this experience, the learning agent attempts to find behavioral policies that improve its expected accumulation of reward. A

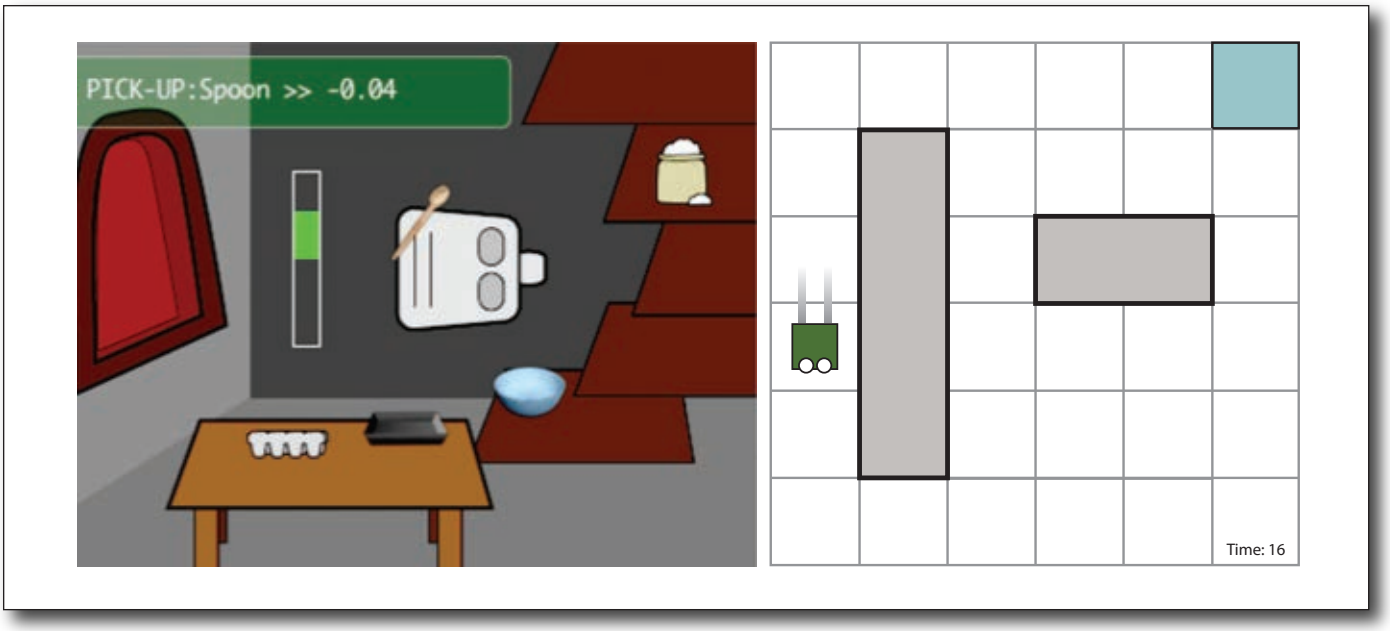


Figure 4. Two Task Domains for Reinforcement Learning Agents Taught by Human Users.

Left: A cooking robot that must pick up and use the ingredients in an acceptable order (Thomaz and Breazeal 2006). The green vertical bar displays positive feedback given by a click-and-drag interface. *Right:* A simulated robot frog that users teach how to navigate to the water (Knox and Stone 2012).

number of research projects have investigated the scenario in which this reward comes as feedback from a human user rather than a function predefined by an expert (Isbell et al. 2006, Thomaz and Breazeal 2008, Knox and Stone 2012). In evaluating the feasibility of nonexpert users teaching through reward signals, these researchers aimed to both leverage human knowledge to improve learning speed and permit users to customize an agent's behavior to fit their own needs.

Thomaz and Breazeal (2008) observed that people have a strong tendency to give more positive rewards than negative rewards. Knox and Stone (2012) later confirmed this positive bias in their own experiments. They further demonstrated that such bias leads many agents to avoid the goal that users are teaching it to reach (for example, the water in figure 4). This undesirable consequence occurs with a common class of reinforcement learning algorithms: agents that value reward accrued over the long term and are being taught to complete so-called episodic tasks. This insight provided justification for the previously popular solution of making agents that hedonistically pursue only short-term human reward, and it led Knox and Stone (2013) to create an algorithm that successfully learns by valuing human reward that can be gained in the long term. Agents trained through their novel approach were more robust to environmental changes and behaved more appropriately in unfamiliar states than did more hedonistic (that is, myopic) variants. These agents and the algorithmic design guidelines Knox and Stone created were the result of multiple iterations of user studies, which identified positive bias and then verified its hypothesized effects.

People Want to Demonstrate How Learners *Should* Behave

In an experiment by Thomaz and Breazeal (2008) users trained a simulated agent to bake a cake through a reinforcement learning framework. In their interface, users gave feedback to the learner by clicking and dragging a mouse — longer drags gave larger-magnitude reward values, and the drag direction determined the valence (+/–) of the reward value (figure 4). Further, users could click on specific objects to signal that the feedback was specific to that object, but they were told that they could not communicate which action the agent should take.

Thomaz and Breazeal found evidence that people nonetheless gave positive feedback to objects that they wanted the agent to manipulate, such as an empty bowl that the agent is in position to pick up. These users violated the instructions by applying what could be considered an irrelevant degree of freedom — giving feedback to objects that had not been recently manipulated — to provide guidance to the agent about future actions, rather than actual feedback about previous actions. After Thomaz and Breazeal adapted the agent's interface and algorithm to incorporate such guidance, the agent's learning performance significantly improved.

Other researchers have reached similar conclusions. In a Wizard-of-Oz study (that is, the agent's outputs were secretly provided by a human) by Kaochar and colleagues (2011), users taught a simu-

lated unmanned aerial vehicle (UAV) to conduct various missions. At any time, these users chose whether to teach by demonstration, by feedback, or by providing an example of a concept. They could also test the agent to see what it had learned. The authors found that users never taught exclusively by feedback, instead generally using it after teaching by the other available means. Together, these two studies provide insight into the design of natural interfaces for teaching agents.

People Naturally Want to Provide More Than Just Data Labels

Labeling data remains the most popular method for end-user input to interactive machine-learning systems because of its simplicity and ease of use. However, as demonstrated in previous case studies, label-based input can have drawbacks (for example, negative attitudes toward being treated as an oracle). In addition, emerging research suggests that in some scenarios users may desire richer control over machine-learning systems than simply labeling data.

For example, Stumpf and colleagues (2007) conducted a study to understand the types of input end users might provide to machine-learning systems if unrestricted by the interface. The authors generated three types of explanations for predictions from a text classification system operating over email messages. These explanations were presented to people in the form of paper-based mockups to avoid the impression of a finished system and encourage people to provide more feedback. People were then asked to give free-form feedback on the paper prototypes with the goal of trying to correct the classifier's mistakes. This experiment generated approximately 500 feedback instances from participants, which were then annotated and categorized. The authors found that people naturally provided a wide variety of input types to improve the classifier's performance, including suggesting alternative features to use, adjusting the importance or weight given to different features, and modifying the information extracted from the text. These results present an opportunity to develop new machine-learning algorithms that might better support the natural feedback people want to provide to learners, rather than force users to interact in limited, learner-centered ways.

People Value Transparency in Learning Systems

In addition to wanting richer controls, people sometimes desire more transparency about how their machine-learning systems work. Kulesza and colleagues (2012) provided users of a content-based music recommender with a 15-minute tutorial discussing how the recommender worked and how various feedback controls (for example, rating songs, steering toward specific feature values, and so on) would affect the learner. Surprisingly, participants responded positively to learning these details about the system. In addition, the researchers found that the more participants learned about the recommender while interacting with it, the more satisfied they were with the recommender's output. This case study provides evidence that users are not always satisfied by "black box" learning systems — sometimes they want to provide nuanced feedback to steer the system, and they are willing and able to learn details about the system to do so.

Examining transparency at a more social level, Rashid and colleagues (2006) examined the effect of showing users the value of their potential movie ratings to a broader community in the MovieLens recommendation system. Users who were given information about the value of their contribution to the entire MovieLens community provided more ratings than those who were not given such information, and those given information about value to a group of users with similar tastes gave more ratings than those given information regarding the full MovieLens community.

Transparency Can Help People Provide Better Labels

Sometimes users make mistakes while labeling, thus providing false information to the learner. Although most learning systems are robust to the occasional human error, Rosenthal and Dey set out to solve this problem at the source. They sought to reduce user mistakes by providing targeted information when a label is requested in an active learning setting. The information provided to the user included a combination of contextual features of the sample to be labeled, explanations of those features, the learner's own prediction of the label for the sample, and its uncertainty in this prediction (Rosenthal and Dey 2010).

They conducted two studies to determine the subset of such information that is most effective in improving the labeling accuracy of users. The first involved people labeling strangers' emails into categories, as well as labeling the interruptability of strangers' activities; the second involved people labeling sensory recordings of their own physical activity. Both studies found that the highest labeling accuracy occurred when the system provided sufficient contextual features and current predictions without uncertainty information. This line of research demonstrates that the way in which information is presented (for example, with or without context) can greatly affect the quality of the response elicited from the user. This case study also shows that not all types of transparency improve the performance of interactive machine-learning systems, and user studies can help determine what information is most helpful to the intended audience.

Summary

Understanding how people actually interact — and want to interact — with machine-learning systems is critical to designing systems that people can use effectively. Exploring interaction techniques through user studies can reveal gaps in a designer's assumptions about their end users and may suggest new algorithmic solutions. In some of the cases we reviewed, people naturally violated assumptions of the machine-learning algorithm or were unwilling to comply with them. Other cases demonstrated that user studies can lead to helpful insights about the types of input and output that interfaces for interactive machine learning should support. In general, this type of research can produce design suggestions and considerations, not only for people building user interfaces and developing the overall user experience, but for the machine-learning community as well.

Novel Interfaces for Interactive Machine Learning

As many of the case studies in the previous section showed, end users often desire richer involvement in the interactive machine-learning process than labeling instances. In addition, research on cost-benefit trade-offs in human-computer interaction has shown that people will invest time and attention into complex tasks if they perceive their efforts to have greater benefits than costs (Blackwell 2002). For example, research on end-user programming has shown that end users program often (for example, through spreadsheets, macros, or mash-ups), but do so primarily to accomplish some larger goal (Blackwell 2002). The act of programming is an investment, and the expected benefit is using the program to accomplish their goal sooner or with less effort than doing it manually. Similarly, this theory suggests that people will invest time to improve their machine learners only if they view the task as more beneficial than costly or risky — that is, when they perceive the benefits of producing an effective learner as outweighing the costs of increased interaction. Therefore, we believe there is an opportunity to explore new, richer interfaces that can leverage human knowledge and capabilities more efficiently and effectively.

In this section, we present case studies that explore novel interfaces for interactive machine-learning systems and demonstrate the feasibility of richer interactions. Interface novelty in these cases can come from new methods for receiving input or providing output. New input techniques can give users more control over the learning system, allowing them to move beyond labeling examples. Such input techniques include methods for feature creation, reweighting of features, adjusting cost matrices, or modifying model parameters. Novel output techniques can make the system's state more transparent or understandable. For example, a system could group unlabeled data to help users label the most informative items, or it could communicate uncertainty about the system's predictions.

These case studies also reinforce our proposition that interactive machine-learning systems should be evaluated with potential end users. Such evaluations are needed both to validate that these systems perform well with real users and to gain insights for further improvement. Many of the novel interfaces detailed in this section were found to be beneficial, but some of the case studies also demonstrate that certain types of input or output may lead to obstacles for the user or reduce the accuracy of the resulting learner. Therefore, novel interfaces should be designed with care and appropriately evaluated before deployment.

Supporting Assessment of Model Quality

In each iteration of the interactive machine-learning process, the user may assess the quality of the current model and then decide how to proceed with further input. A common technique for conveying model quality in supervised learning is to present a person with all of the unlabeled data sorted by their predicted scores for some class (for example, classification probabilities or relevance rankings). After evaluating this presentation, a person then decides how to proceed in training by selecting additional examples to label for further input. Although straightforward, this technique inefficiently illustrates learner quality and provides the user with no guidance in selecting additional training examples.

Fogarty and colleagues (2008) investigated novel techniques for presenting model quality in CueFlik, an interactive machine-learning system for image classification (figure 5). Through a user study, the authors demonstrated that a technique of presenting users with only the best- and worst-matching examples enabled users to evaluate model quality more quickly and, in turn, train significantly better models than the standard technique of presenting the user with all of the data. In a follow-up investigation with CueFlik, Amershi and colleagues (2009) went on to show that presentation techniques designed to summarize model quality for users while providing them with high-value examples to choose from as further input to the model led users to train better models than the best- and worst-matching technique from previous work. These case studies demonstrate that presentation matters and designing interfaces that balance the needs of both end users and machine learners is more effective than optimizing user interfaces for end users in isolation.

Supporting Experimentation with Model Inputs

Interactive machine learning enables rapid and incremental iterations between the end user and the

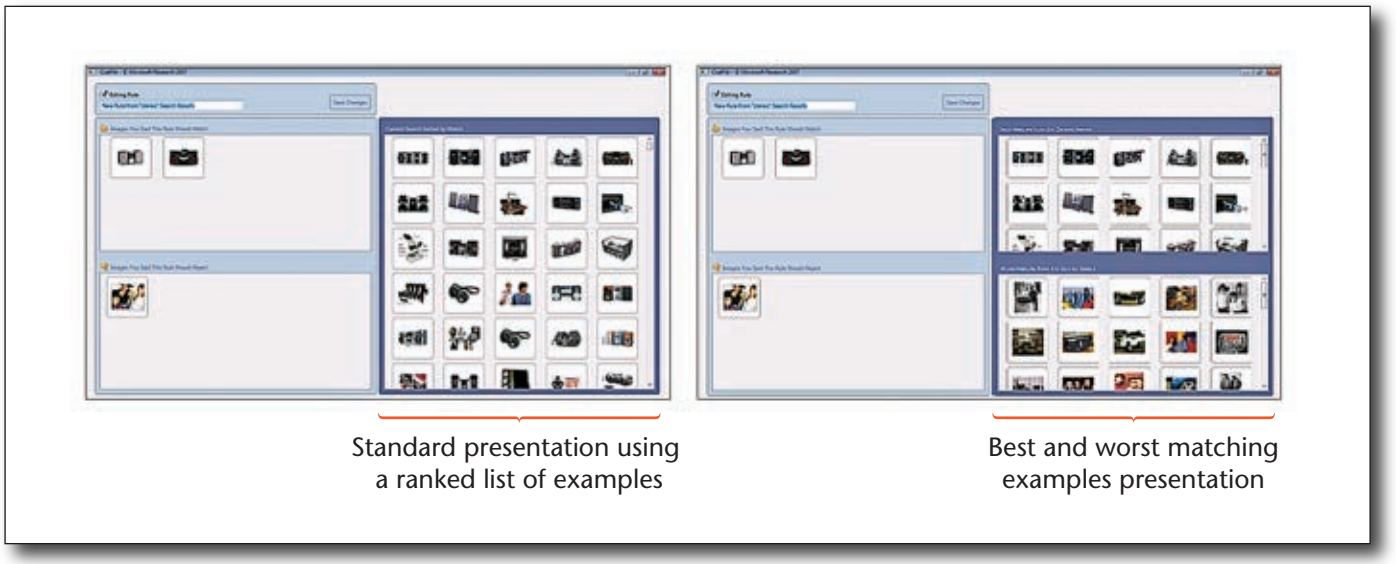


Figure 5. Comparing the Quality of a Machine-Learned Visual Concept.

Fogarty et al.'s work (2008) with CueFlik compared two methods of illustrating the quality of a machine-learned visual concept.The standard method (left) presented users with examples ranked by their likelihood of membership to the positive class. The best and worst matches method (right) instead showed examples predicted as positive or negative with high certainty by CueFlik. A user study showed that the best- and worst-matches technique led users to train significantly better learners than the standard presentation.

machine learner. As a result, users may want to experiment with alternative inputs and examine resulting model outputs before committing to any model input. To support end-user experimentation, Amershi and colleagues (2010) augmented the CueFlik system (figure 6) discussed previously with a history visualization to facilitate model comparison and support for model revision (through undo or redo, removing labels, and reverting to previous models using the history visualization). In a user study, Amershi et al. showed that end users used revision when it was available and this led them to achieve better final models in the same amount of time (even while performing more actions) compared to when these supports were unavailable. Furthermore, being able to examine and revise actions is consistent with how people expect to interact with their applications. One participant in this study commented that without revision "it felt a little like typing on a keyboard without a backspace key." This case study illustrates that end users may be willing and may expect options to experiment and revise their inputs to machine learners during the interactive machine-learning process.

Appropriately Timing Queries to the User

As discussed earlier, applying active learning to interactive settings can be undesirable to the user when questions come in a constant stream from the learning system. To address this problem, Cakmak and Thomaz (2010) proposed intermittently active learning, in which only a subset of the examples provided by the user are obtained through queries. This brings a new challenge for the learner: deciding when to query as opposed to letting the user choose an example. Cakmak and Thomaz explored two approaches. In the first, the learner made queries only when certain conditions were met. It took into account the quality of examples chosen by the user and the probability that the user could randomly provide useful examples. In the second approach, the user decided when the learner was allowed to ask questions (that is, a query was made only when the user said "do you have any questions?").

A study comparing intermittently active learning with fully active and passive learning demonstrated its advantage over these two extremes (Cakmak, Chao, and Tomaz 2010). The study showed that both intermittent approaches resulted in learning as fast as the fully active approach, while being subjectively preferred over fully active or fully passive approaches. The interactions with the intermittently active learners were found to be more balanced, enjoyable, and less frustrating. When asked to choose between the two alternative approaches, users preferred the teacher-triggered queries, mentioning that they liked having full control over the learner's queries. As exemplified in this case study, building interactive learning systems that fit user preferences can sometimes require the modification of existing methods in fundamental ways.

Enabling Users to Query the Learner

In addition to the learner querying the user as in the

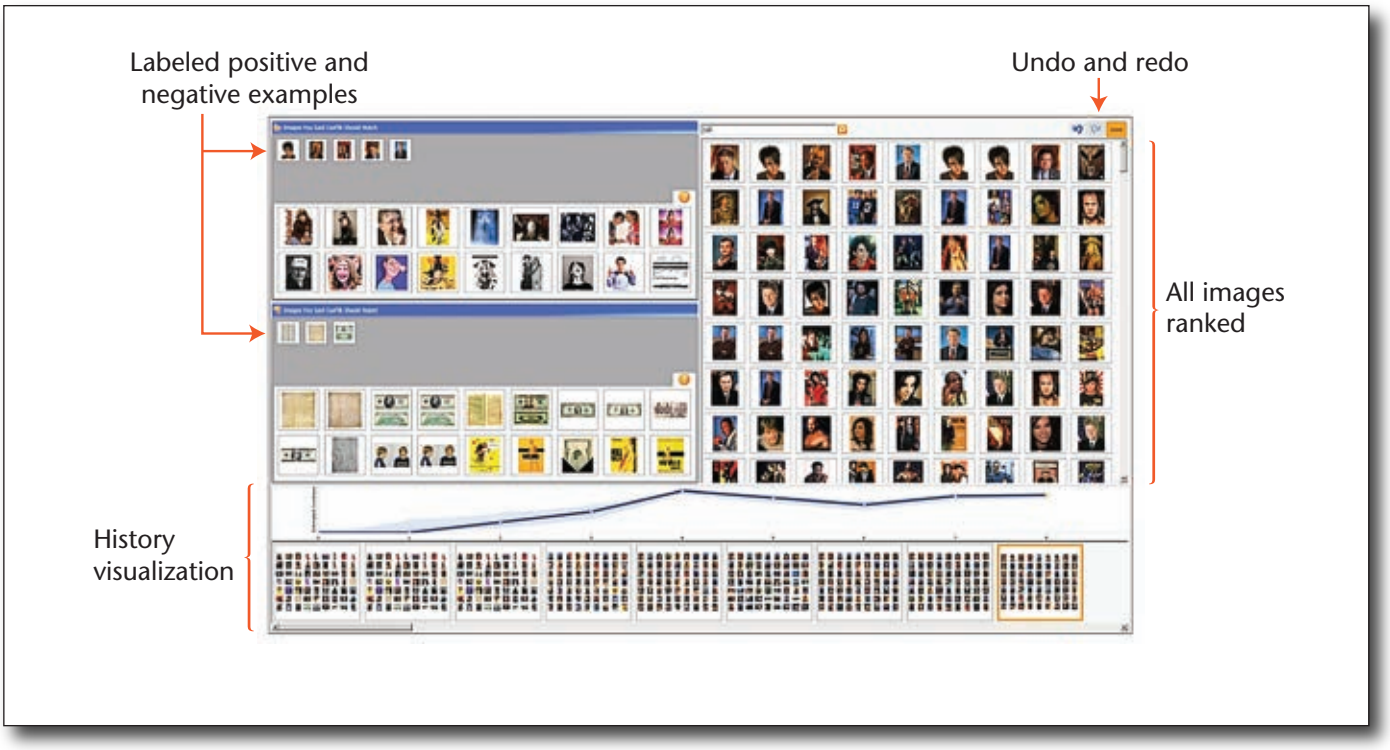


Figure 6. CueFlik.

CueFlik augmented with a history visualization to facilitate model comparison and support for model revision. Amershi et al. (2010) showed that these supports for experimentation during interactive machine learning enabled end users to train better quality models than when these supports were unavailable.

active learning paradigm, sometimes the user may want to query the learner. Kulesza and colleagues (2011) developed an approach to let users ask a text classifier why it was behaving in a particular way (for example, "Why was this classified as *X* instead of *Y*?"). The learner's responses were interactive, thus providing a way for users not only to understand why the system had made a particular prediction, but also to adjust the learner's reasoning if its prediction was wrong. For example, the learner could display a bar graph showing that it associated the word "job" with the topic of "news" more than the topic of "résumés"; if the user disagreed with this reasoning, he or she could adjust the graph to tell the learner that "jobs" should be associated with "résumés" more than "news".

Most participants exposed to this why-oriented interaction approach significantly increased the accuracy of their naïve Bayes text classifiers; however, every participant encountered a number of barriers while doing so. In particular, participants had trouble selecting features to modify from the thousands in the bag-of-words feature set. Also, once participants did select features to adjust, they had trouble understanding how changes to a single feature altered the learner's predictions for seemingly unrelated items. This study suggests that for learners with large feature sets or complex interactions between features, users will need additional support to make sense of which features are most responsible for an item's classification.

Enabling Users to Critique Learner Output

Some machine-learning systems help users navigate an otherwise unnavigable search space. For example, recommender systems help people find specific items of interest, filtering out irrelevant items. Vig, Sen, and Riedl (2011) studied a common problem in this domain: recommending results that are close, but not quite close enough, to what the user was looking for. Researchers developed a prototype to support tag-based "critiques" of movie recommendations. Users could respond to each recommendation with refinements such as "Like this, but less violent" or "Like this, but more cerebral," where violent and cerebral are tags that users had applied to various movies. A *k*-nearest-neighbor approach was then used to find similar items that included the user-specified tags.

This relatively simple addition to the MovieLens website garnered an overwhelmingly positive reaction, with 89 percent of participants in a user study saying that they liked it, and 79 percent requesting that it remain a permanent feature on the site. This example helps illustrate both the latent desire among users for better control over machine-learn-

Figure 7. ManiMatrix System.

The ManiMatrix system displays the confusion matrix of the classifier and allows the user to directly increase or decrease the different types of errors using arrows on the matrix cells. ManiMatrix provides feedback to the user by highlighting cells that change value as a result of the user's click (red indicates a decrease and green indicates an increase).

ing systems and that, by supporting such control in an interactive fashion, user attitudes toward the learner can be greatly enhanced.

Allowing Users to Specify Preferences on Errors

People sometimes want to refine the decision boundaries of their learners. In particular, for some classifiers it might be critical to detect certain classes correctly, while tolerating errors in other classes (for example, misclassifying spam as regular email is typically less costly than misclassifying regular email as spam). However, refining classifier decision boundaries is a complex process even for experts, involving iterative parameter tweaking, retraining, and evaluation. This is particularly difficult because among parameters there are often dependencies that lead to complex mappings between parameter values and the behavior of the system.

To address these difficulties, Kapoor and colleagues (2010) created ManiMatrix (figure 7), a tool for people to specify their preferences on decision boundaries through interactively manipulating a classifier's confusion matrix (that is, a breakdown of the correct and incorrect predictions it made for each class). Given these preferences, ManiMatrix employs Bayesian decision theory to compute decision boundaries that minimize the expected cost of different types of errors, and then visualizes the results for further user refinement. A user study with machine-learning novices demonstrated that participants were able to quickly and effectively modify decision boundaries as desired with the ManiMatrix system. This case study demonstrates that nonexperts can directly manipulate a model's learning objective, a distinctly different form of input than choosing examples and labeling them.

Combining Models

An ensemble classifier is a classifier that builds its prediction from the predictions of multiple subclassifiers, each of which are functions over the same space as the ensemble classifier. Such ensembles often outperform all of their subclassifiers and are a staple of applied machine learning (for example, AdaBoost by Freund and Schapire [1995]). A common workflow for creating ensemble classifiers is to experiment with different features, parameters, and algorithms through trial and error or hill-climbing through the model space. Even for machine-learning experts, however, this approach can be inefficient and lead to suboptimal performance.

To facilitate the creation of ensemble classifiers, Talbot and colleagues (2009) developed EnsembleMatrix, a novel tool for helping people interactively build, evaluate, and explore different ensembles (figure 8). EnsembleMatrix visualizes the current ensemble of individual learners through a confusion matrix. The user can then experiment with and evaluate different linear combinations of individual learners by interactively adjusting the weights of all models through a single two-dimensional interpolation widget (top right in figure 8). EnsembleMatrix's novel interface also allows people to make use of their visual processing capabilities to partition the confusion matrix according to its illustrated performance, effectively splitting the ensemble into subensembles that can be further refined as necessary. A user study showed that EnsembleMatrix enabled people to create ensemble classifiers on par with the best published ensembles on the same data set. Furthermore, they managed to do so in a single, one-hour session. The study involved participants ranging from machine-learning novices to experts.

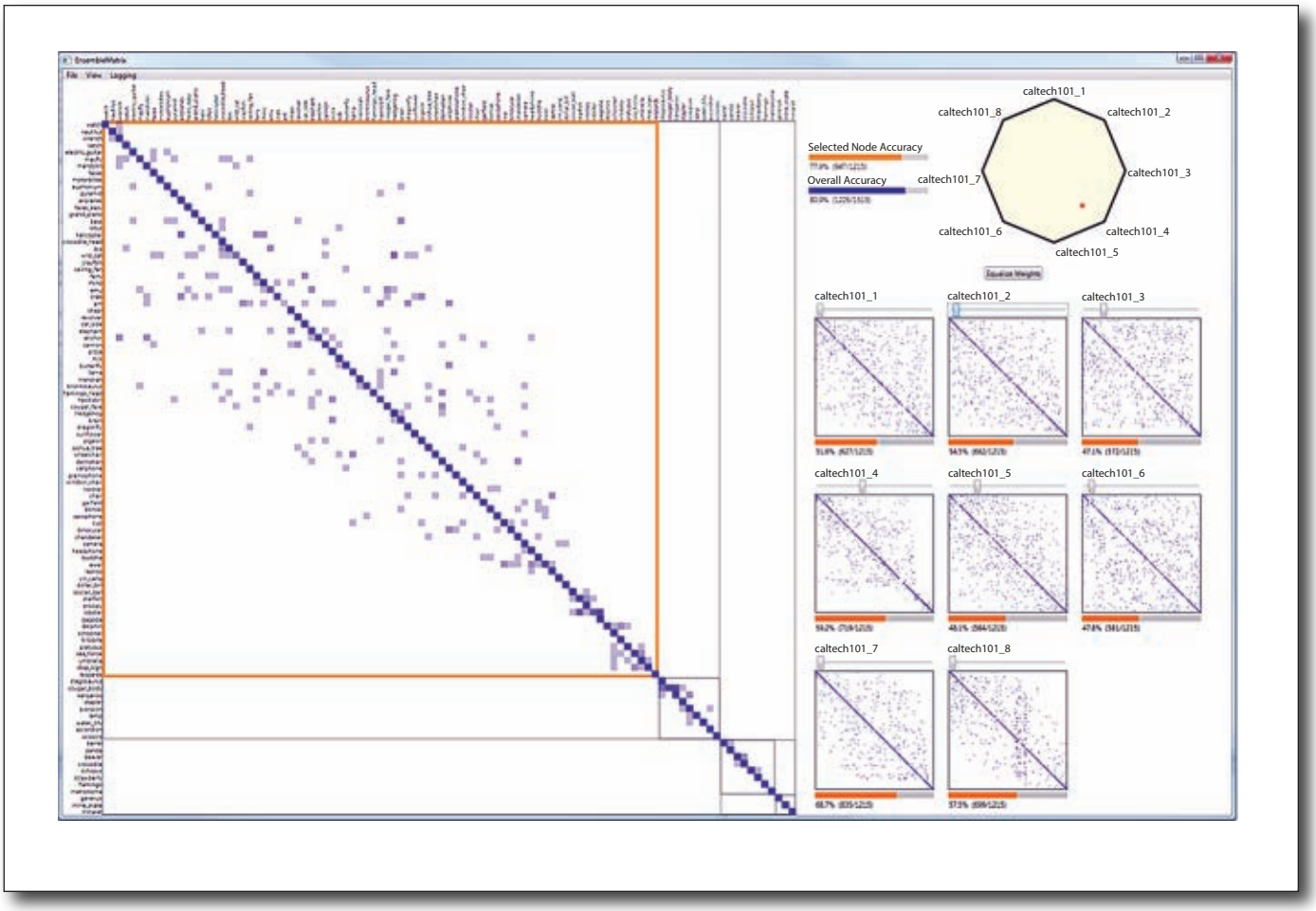


Figure 8. EnsembleMatrix.

EnsembleMatrix visualizes the current ensemble (left) of individual learners (bottom right) through a confusion matrix. Users can adjust the weights of individual models through a linear combination widget (top right) to experiment with different ensembles. Users can also partition the confusion matrix to split and refine subensembles.

This case study illustrates that effectively combining human intuition and input with machine processing can enable people to create better classifiers in less time than standard approaches that ignore these powerful human capabilities.

Summary

Whether a new interface will improve the user's experience or the system's performance can only be assessed through evaluation with potential end users. In the case studies above, permitting richer user interactions was often beneficial, but not always so. Different users have different needs and expectations of the systems they employ. In addition, rich interaction techniques may be appropriate for some scenarios and not others. Thus, conducting user studies of novel interactive machine-learning systems is critical not only for discovering promising modes of interaction, but also to uncover obstacles that users may encounter in different scenarios and unspoken assumptions they might hold about machine learners. The accumulation of such research can facilitate the development of design guidelines for building future interactive machine-learning systems, much like those that exist for traditional software systems (for example, Shneiderman et al. [2009]).

Discussion

Interactive machine learning is a potentially powerful technique for enabling end-user interaction with machine learning. As this article illustrates, studying how people interact with interactive machine-learning systems and exploring new techniques for enabling those interactions can result in better user experiences and more effective machine learners. However, research in this area has only just begun, and many opportunities remain to improve the interactive machine-learning process. This section

describes open challenges and opportunities for advancing the state of the art in human interaction with interactive machine-learning systems.

Developing a Common Language Across Diverse Fields

As shown by the variety of case studies presented in this article, many fields of computer science already employ interactive machine learning to solve different problems, such as search in information retrieval, filtering in recommender systems, and task learning in human-robot interaction. However, different fields often refer to interactive machine learning or parts of the interactive machine-learning process in domain-specific terms (for example, relevance feedback, programming by demonstration, debugging machine-learned programs, socially guided machine learning). This diversity in terminology impedes awareness of progress in this common space and can potentially lead to duplicate work. Seeking to develop a common language and facilitate the development of new interactive machine-learning systems, some researchers have begun to examine this body of work and abstract away domain-specific details from existing solutions to characterize common variables and dimensions of the interactive machine-learning process itself (for example, Amershi [2012]; Porter, Theiler, and Hush [2013]).

For example, Amershi (2012) examined interactive machine-learning systems across several fields (including information retrieval, context-aware computing, and adaptive and intelligent systems) and identified specific design factors influencing human interaction with machine-learning systems (for example, the expected duration of model use, the focus of a person's attention during interaction, the source and type of data over which the machine will learn) and design dimensions that can be varied to address these factors (for example, the type and visibility of model feedback, the granularity and direction of user control, and the timing and memory of model input). In another example, Porter, Theiler, and Hush (2013) break down the interactive machine-learning process into three dimensions: task decomposition (defining the level of coordination and division of labor between the end user and the machine learner), training vocabulary (defining the type of input end users can provide the machine learner), and the training dialogue (defining the level and frequency of interaction between the end user and the learner). Design spaces such as these can help to form a common language for researchers and developers to communicate new interactive machine-learning solutions and share ideas. However, there are many ways to dissect and describe the various interaction points between people and machine learners within the interactive machine-learning process. Therefore, an important opportunity remains for converging on and adopting a common language across these fields to help accelerate research and development in this space.

Distilling Principles and Guidelines for How to Design Human Interaction with Machine Learning

In addition to developing a common language, an opportunity remains for generalizing from existing solutions and distilling principles and guidelines for how we should design future human interaction with interactive machine learning, much like we have for designing traditional interfaces (for example, Schneiderman et al. [2009]; Moggridge and Smith [2007]; Dix et al. [2004]; Winograd [1996]; Norman [1988]). For example, Schneiderman's golden rules of interface design advocate for designating the users as the controllers of the system and offering them informative feedback after each interaction.

Some principles for designing traditional interfaces can directly translate to the design of interactive machine learning interfaces — interactive machine-learning systems inherently provide users with feedback about their actions and, as this article discusses, giving users more control over machine-learning systems can often improve a user's experience. However, interactive machine-learning systems also often inherently violate many existing interface design principles. For example, research has shown that traditional interfaces that support understandability (that is, systems that are predictable or clear about how they work) and actionability (that is, systems that make it clear how a person can accomplish his or her goals and give the person the freedom to do so) are generally more usable than interfaces that do not support these principles. Many machine-learning systems violate both principles: they are inherently difficult for users to understand fully and they largely limit the control given to the end user. Thus, there is an opportunity to explore how current design principles apply to the human-computer interaction in interactive machine learning.

Some researchers have started to suggest new principles for designing end-user interaction with general artificial intelligence systems, many of which could translate to end-user interaction with interactive machine learning (for example, Norman [1994]; Höök [2000]; Horvitz [1999]; Jameson [2009]). For example, Norman (1994) and Höök (2000) both identified safety and trust as key factors to consider when designing intelligent systems, referring to the assurance against and prevention of unwanted adaptations or actions. Others have stated that artificially intelligent and machine-learning-based systems should manage expectations to avoid misleading or frustrating the user during interaction (for example, Norman [1994]; Höök [2000]; Jameson [2009]). In Horvitz's formative paper on mixed-initiative inter-

faces (1999), he proposed several principles for balancing artificial intelligence with traditional direct-manipulation constructs. For example, Horvitz emphasized consideration of the timing of interactive intelligent services, limiting the scope of adaptation or favoring direct control under severe uncertainty, and maintaining a working memory of recent interactions. While these suggestions can help guide the design of future systems, more work remains to develop a comprehensive set of guidelines and principles that work in various settings. Often such design principles are distilled from years of experience developing such interactions. Alternatively, we may accelerate the development of such guidelines by extracting dimensions that can be manipulated to design interactive machine-learning systems and systematically evaluating general solutions in varying settings.

Developing Techniques and Standards for Appropriately Evaluating Interactive Machine-Learning Systems

Although systematic evaluation can facilitate generalization and transfer of ideas across fields, the interleaving of human interaction and machine-learning algorithms makes reductive study of design elements difficult. For example, it is often difficult to tease apart whether failures of proposed solutions are due to limitations of the particular interface or interaction strategies used, the particular algorithm chosen, or the combination of the interaction strategy with the particular algorithm used. Likewise, inappropriately attributing success or failure to individual attributes of interactive machine-learning solutions can be misleading. Therefore, new evaluation techniques may be necessary to appropriately gauge the effectiveness of new interactive machine-learning systems. In addition, as our case studies illustrated, some interaction techniques may be appropriate for certain scenarios of use but not others. Evaluations should therefore be careful not to overgeneralize successes or failures of specific interaction techniques. Rather, the scenarios and contexts of use should be generalized to better understand when to apply certain techniques over others.

Leveraging the Masses During Interaction with Machine Learning

Most of the case studies in this article focused on a single end user interacting with a single machine-learning system. However, the increasing proliferation of networked communities and crowd-powered systems provides evidence of the power of the masses to collaborate and produce content. An important opportunity exists to investigate how crowds of people might collaboratively drive interactive machine-learning systems, potentially scaling up the impact of such systems. For example, as interactive machine learning becomes more prevalent in our everyday applications, people should be able to share and reuse machine learners rather than have to start from scratch. Moreover, people should be able to bootstrap, build upon, and combine learners to configure more sophisticated data processing and manipulation. A few have started to explore such opportunities (for example, Hoffman et al. [2009]; Kamar, Hacker, and Horvitz [2012]; Law and von Ahn [2009]), but more work remains to fully understand the potential of multiple end users interacting with machine-learning systems. For example, work remains in understanding how people can meaningfully describe, compare, and search for existing machine learners in order to build upon them, in understanding how learners can be generalized or transformed for new situations and purposes, in understanding how we can create composable learners to enable more powerful automation, and in understanding how we can coordinate the efforts of multiple people interacting with machine-learning systems.

Algorithmic Problems in Interactive Machine Learning

Research on user interactions with interactive machine learning raises two important technical challenges. First, the requirement for rapid model updates often necessitates trading off accuracy with speed. The resulting models are therefore suboptimal. Although interactive machine learning can deal with this problem through more iterations, algorithms that are both fast and accurate would improve the quality of learned models and reduce the number of iterations needed to obtain useful models. Second, as some of the case studies described in this article showed, users may desire to interact with machine-learning systems in ways that are not supported by existing machine-learning methods. Addressing this challenge requires the development of new frameworks and algorithms that can handle different inputs and outputs that are desirable and natural for end users.

Increasing Collaboration Across the Fields of Human Computer Interaction and Machine Learning

The inherent coupling of the human and machine in interactive machine learning underscores the need for collaboration across the fields of human-computer interaction and machine learning. This collaboration will benefit human-computer interaction researchers in solving the algorithmic problems discussed above and provide more powerful tools to end users. In turn, machine-learning researchers would benefit by having new methods evaluated with potential users to address practical issues and by developing new frameworks that support realistic assumptions about users.

Finally, we believe that the diversity of perspectives will benefit both communities. For example, when dealing with noisy problems, machine-learning researchers have often attempted to develop algorithms that work despite the noise, whereas human-computer interaction researchers often try to develop interaction techniques to reduce the noise that end users induce. Collaboration between these two communities could leverage the benefits of both solutions.

Conclusion

The case studies presented in this article support three key points. First, interactive machine learning differs from traditional machine learning. The interaction cycles in interactive machine learning are typically more

rapid, focused, and incremental than in traditional machine learning. This increases the opportunities for users to affect the learner and, in turn, for the learner to affect the users. As a result, the contributions of the system and the user to the final outcome cannot be decoupled, necessitating an increased need to study the system together with its potential users.

Second, explicitly studying the users of learning systems is critical to advancing this field. Formative user studies can help identify user needs and desires, and inspire new ways in which users could interact with machine-learning systems. User studies that evaluate interactive machine-learning systems can reveal false assumptions about potential users and common patterns in their interaction with the system. User studies can also help to identify common barriers faced by users when novel interfaces are introduced.

Finally, the interaction between learning systems and their users need not be limited. We can build powerful interactive machine-learning systems by giving more control to end users than the ability to label instances, and by providing users with more transparency than just the learner's predicted outputs. However, more control for the user and more transparency from the learner do not automatically result in better systems, and in some situations may not be appropriate or desired by end users. We must continue to evaluate novel interaction methods with real users to understand whether they help or hinder users' goals.

In addition to demonstrating the importance and potential of research in interactive machine learning, this article characterized some of the challenges and opportunities that currently confront this field. By acknowledging and embracing these challenges, we can move the field of interactive machine learning forward toward more effective interactions. We believe this will lead not only to more capable machine learners, but also more capable end users.

Notes

1. All authors contributed equally to this article.

2. In this article we examine interactive machine-learning systems in which the human is consciously interacting with the machine learner in order to improve it. That is, we do not consider interactive machine-learning systems that obtain user feedback implicitly (for example, websites that may automatically adapt their presentation to a user's click history without the user's knowledge).

References

Amershi, S. 2012. Designing for Effective End-User Interaction with Machine Learning. Ph.D. Dissertation, Department of Computer Science. University of Washington, Seattle, WA. (hdl.handle.net/1773/22006)

Amershi, S.; Cakmak, M.; Knox, W. B.; Kulesza, T.; and Lau, T. 2013. IUI Workshop on Interactive Machine Learning. In *Proceedings of the Companion Publication of the 2013 International Conference on Intelligent User Interfaces,* 121–124. New York: Association for Computing Machinery. dx.doi.org/10.1145/2451176.2451230

Amershi, S.; Fogarty, J.; Kapoor, A.; and Tan, D. 2010. Examining Multiple Potential Models in End-User Interactive Concept Learning. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* 2010 (CHI 2010), 1357–1360. New York: Association for Computing Machinery.

Amershi, S.; Fogarty, J.; Kapoor, A.; and Tan, D. 2009. Overview-Based Example Selection in Mixed-Initiative Concept Learning. In *Proceedings of the ACM Symposium on User Interface Software and Technology,* 2009 (UIST 2009), 47–256. New York: Association for Computing Machinery.

Blackwell, A. F. 2002. First Steps in Programming: A Rationale for Attention Investment Models. In *Proceedings of the 2002 IEEE Symposium on Human Centric Computing Languages and Environments,* 2–10. Piscataway, NJ: Institute for Electrical and Electronics Engineers.

Cakmak, M., and Thomaz, A. L. 2010. Optimality of Human Teachers for Robot Learners. In *Proceedings of the 2010 IEEE 9th International Conference on Development and Learning* (ICDL), 64–69. Piscataway, NJ: Institute for Electrical and Electronics Engineers.

Cakmak, M.; Chao, C.; and Thomaz, A. L. 2010. Designing Interactions for Robot Active Learners. *Autonomous Mental Development* 2(2): 108–118. dx.doi.org/10.1109/TAMD.2010.2051030

Caruana, R.; Elhaway, M.; Nguyen, N.; and Smith, C. 2006. Meta Clustering. In *Proceedings of the Sixth IEEE International Conference on Data Mining,* 2006. (ICDM'06), 107–118. Piscataway, NJ: Institute for Electrical and Electronics Engineers.

Cohn, D.; Caruana, R.; and McCallum, A. 2003. Semi-Supervised Clustering with User Feedback. *Constrained Clustering: Advances in Algorithms, Theory, and Applications* 4(1): 17–32.

Dix, A.; Finlay, J.; Abowd, G. D.; and Beal, R. 2004. Interaction Design Basics. In *Human Computer Interaction,* 3rd edition, chapter 5, 189–224. Harlow, England: Pearson Education Ltd.

Fails, J. A., and Olsen Jr, D. R. 2003. Interactive Machine Learning. In *Proceedings of the 8th International Conference on Intelligent User Interfaces,* 39–45. New York: Association for Computing Machinery.

Fiebrink, R.; Cook, P. R.; and Trueman, D. 2011. Human Model Evaluation in Interactive Supervised Learning. In *Proceedings of the Conference on Human Factors in Computing Systems* (CHI 2011), 147–156. New York: Association for Computing Machinery. dx.doi.org/10.1145/1978942.1978965

Fogarty, J.; Tan, D.; Kapoor, A.; and Winder, S. 2008. CueFlik: Interactive Concept Learning in Image Search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* 29–38. New York: Association for Computing Machinery.

Freund, Y., and Schapire, R. E. (1995). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. In *Proceedings of the Second European Conference on Computational Learning Theory,* 23–37. Berlin, Heidelberg: Springer. dx.doi.org/10.1007/3-540-59119-2_166

Guillory, A., and Bilmes, J. A. 2011. Simultaneous Learning and Covering with Adversarial Noise. In *Proceedings of the 28th International Conference on Machine Learning* (ICML-11), 369–376). Princeton, NJ: International Machine Learning Society, Inc.

Hoffman R.; Amershi, S.; Patel, K.; Wu, F.; Fogarty, J.; and Weld, D. S. 2009. Amplifying Community Content Creation with Mixed-Initiative Information Extraction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI 2009), 1849–1858. New York: Association for Computing Machinery.

Höök, K. 2000. Steps to Take Before Intelligent User Interfaces Become Real. *Interacting with Computers* 12(4): 409–426. dx.doi.org/10.1016/S0953-5438(99)00006-5

Horvitz, E. 1999. Principles of Mixed-Initiative User Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* 159–166. New York: Association for Computing Machinery.

Isbell Jr., C. L.; Kearns, M.; Singh, S.; Shelton, C. R.; Stone, P.; and Kormann, D. 2006. Cobot in LambdaMOO: An Adaptive

Social Statistics Agent. *Autonomous Agents and Multi-Agent Systems* 13(3): 327–354. dx.doi.org/10.1007/s10458-006-0005-z

Jameson, A. 2009. Adaptive Interfaces and Agents. *Human-Computer Interaction: Design Issues, Solutions, and Applications,* 105. Boca Raton, FL: CRC Press.

Kamar, E.; Hacker, S.; and Horvitz, E. 2012. Combining Human and Machine Intelligence in Large-Scale Crowdsourcing. In *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems* (AAMAS 2012). Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.

Kaochar, T.; Peralta, R. T.; Morrison, C. T.; Fasel, I. R.; Walsh, T. J.; and Cohen, P. R. 2011. Towards Understanding How Humans Teach Robots. In *User Modeling, Adaption and Personalization,* Lecture Notes in Computer Science Volume 6787, 347–352. Berlin: Springer. dx.doi.org/10.1007/978-3-642-22362-4_31

Kapoor, A.; Lee, B.; Tan, D.; and Horvitz, E. 2010. Interactive Optimization for Steering Machine Classification. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 1343–1352. New York: Association for Computing Machinery.

Knox, W. B., and Stone, P. 2013. Learning Non-Myopically from Human-Generated Reward. *In Proceedings of the 2013 International Conference on Intelligent User Interfaces,* 191–202. New York: Association for Computing Machinery. dx.doi.org/10.1145/2449396.2449422

Knox, W. B., and Stone, P. 2012. Reinforcement Learning from Human Reward: Discounting in Episodic Tasks. In *Proceedings of the 2012 IEEE International Workshop on Robots and Human Interactive Communications* (RO-MAN), 878–885. Piscataway, NJ: Institute for Electrical and Electronics Engineers.

Kulesza, T.; Stumpf, S.; Burnett, M.; and Kwan, I. 2012. Tell Me More?: The Effects of Mental Model Soundness on Personalizing an Intelligent Agent. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* 1–10. New York: Association for Computing Machinery.

Kulesza, T.; Stumpf, S.; Wong, W. K.; Burnett, M. M.; Perona, S.; Ko, A.; and Oberst, I. 2011. Why-Oriented End-User Debugging of Naive Bayes Text Classification. *ACM Transactions on Interactive Intelligent Systems* 1(1): 2. dx.doi.org/10.1145/2030365.2030367

Law, E., and von Ahn, R. 2009. Input-Agreement: A New Mechanism for Data Collection Using Human Computation Games. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI 2009). New York: Association for Computing Machinery.

Moggridge, B., and Smith, G. C. 2007. *Designing Interactions*, Volume 17. Cambridge, MA: The MIT Press.

Norman, D. A. 1994. How Might People Interact with Agents. *Communications of the ACM* 37(7): 68–71. dx.doi.org/10.1145/176789.176796

Norman, D. A. 1988. The Design of Everyday Things. New York: Basic Books.

Porter, R.; Theiler, J.; and Hush, D. 2013. Interactive Machine Learning in Data Exploitation. Technical Report. Los Alamos National Laboratories, Los Alamos, NM. dx.doi.org/10.2172/1060903

Pu, P., and Chen, L. 2009. User-Involved Preference Elicitation for Product Search and Recommender Systems. *AI Magazine* 29(4): 93.

Rashid, A. M.; Ling, K.; Tassone, R. D.; Resnick, P.; Kraut, R.; and Riedl, J. 2006. Motivating Participation by Displaying the Value of Contribution. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* 955–958. New York: Association for Computing Machinery.

Rosenthal, S. L., and Dey, A. K. 2010. Towards Maximizing the Accuracy of Human-Labeled Sensor Data. In *Proceedings of the 15th International Conference on Intelligent User Interfaces,* 259–268. New York: Association for Computing Machinery. dx.doi.org/10.1145/1719970.1720006

Settles, B. 2010. Active Learning Literature Survey. Technical Report, Department of Computer Science, University of Wisconsin, Madison, Madison, WI.

Shneiderman, B.; Plaisant, C.; Cohen, M.; and Jacobs, S. 2009. *Designing the User Interface: Strategies for Effective Human-Computer Interaction,* 5th edition. Reading, MA: Addison-Wesley.

Stumpf, S.; Rajaram, V.; Li, L.; Burnett, M.; Dietterich, T.; Sullivan, E.; Drummond, R.; and Herlocker, J. 2007. Toward Harnessing User Feedback for Machine Learning. In *Proceedings of the 12th International Conference on Intelligent User Interfaces,* 82–91. New York: Association for Computing Machinery. dx.doi.org/10.1145/1216295.1216316

Talbot, J.; Lee, B.; Kapoor, A.; and Tan, D. S. 2009. EnsembleMatrix: Interactive Visualization to Support Machine Learning with Multiple Classifiers. In *Proceedings of the 27th International Conference on Human Factors in Computing Systems,* 1283–1292. New York: Association for Computing Machinery.

Thomaz, A. L., and Breazeal, C. 2008. Teachable Robots: Understanding Human Teaching Behavior to Build More Effective Robot Learners. *Artificial Intelligence* 172(6): 716–737.

Vig, J.; Sen, S.; and Riedl, J. 2011. Navigating the Tag Genome. In *Proceedings of the 16th International Conference on Intelligent User Interfaces*, 93–102. New York: Association for Computing Machinery. dx.doi.org/10.1145/1943403.1943418

Winograd, T. 1996. *Bringing Design to Software.* New York: Association for Computing Machinery.

Saleema Amershi is a researcher in the Computer Human Interactive Learning (CHIL) group at Microsoft Research. Her research lies at the intersection of human-computer interaction and machine learning. In particular, her work involves designing and developing new techniques to support effective end-user interaction with interactive machine-learning systems. Amershi received her Ph.D. in computer science from the University of Washington's Computer Science and Engineering Department in 2012.

Maya Cakmak is an assistant professor at the University of Washington, Computer Science and Engineering Department, where she directs the Human-Centered Robotics lab. She received her Ph.D. in robotics from the Georgia Institute of Technology in 2012. Her research interests are in human-robot interaction and end-user programming. Her work aims to develop assistive robots that can be programmed and controlled by end users, in the context of use.

W. Bradley Knox recently completed a postdoctoral researcher position at the MIT Media Lab. His research interests span machine learning, human-robot interaction, and psychology, especially machine-learning algorithms that learn through human interaction. Knox received a Ph.D. in computer science at the University of Texas at Austin and a BS in psychology from Texas A&M University.

Todd Kulesza is a computer science Ph.D. candidate at Oregon State University, working under the guidance of Margaret Burnett. His research interests are in human interactions with intelligent systems, with a focus on enabling end users to personalize such systems efficiently and effectively.

The Dialog State Tracking Challenge Series

Jason D. Williams, Matthew Henderson, Antoine Raux, Blaise Thomson, Alan Black, Deepak Ramachandran

■ *In spoken dialog systems, dialog state tracking refers to the task of correctly inferring the user's goal at a given turn, given all of the dialog history up to that turn. The Dialog State Tracking Challenge is a research community challenge task that has run for three rounds. The challenge has given rise to a host of new methods for dialog state tracking and also to deeper understanding about the problem itself, including methods for evaluation.*

Conversational systems are increasingly becoming a part of daily life, with examples including Apple's Siri, Google Now, Nuance Dragon Go, Xbox, and Cortana from Microsoft, and those from numerous startups. In the core of a conversation system is a key component called a dialog state tracker, which estimates the user's goal given all of the dialog history so far. For example, in a tourist information system, the dialog state might indicate the type of business the user is searching for (pub, restaurant, coffee shop), the desired price range, and the type of food served. Dialog state tracking is difficult because automatic speech recognition (ASR) and spoken language understanding (SLU) errors are common and can cause the system to misunderstand the user. At the same time, state tracking is crucial because the system relies on the estimated dialog state to choose actions — for example, which restaurants to suggest. Figure 1 shows an illustration of the dialog state tracking task.

Historically dialog state tracking has been done with hand-crafted rules. More recently, statistical methods have been found to be superior by effectively overcoming some SLU errors, resulting in better dialogs. Despite this progress, direct comparisons between methods have not been possible because past studies use different domains, system components, and evaluation measures, hindering progresss. The Dialog State Tracking Challenge (DSTC) was initiated to address this barrier by providing a common test bed and evaluation framework for dialog state tracking algorithms.

 ISSN 0738-4602

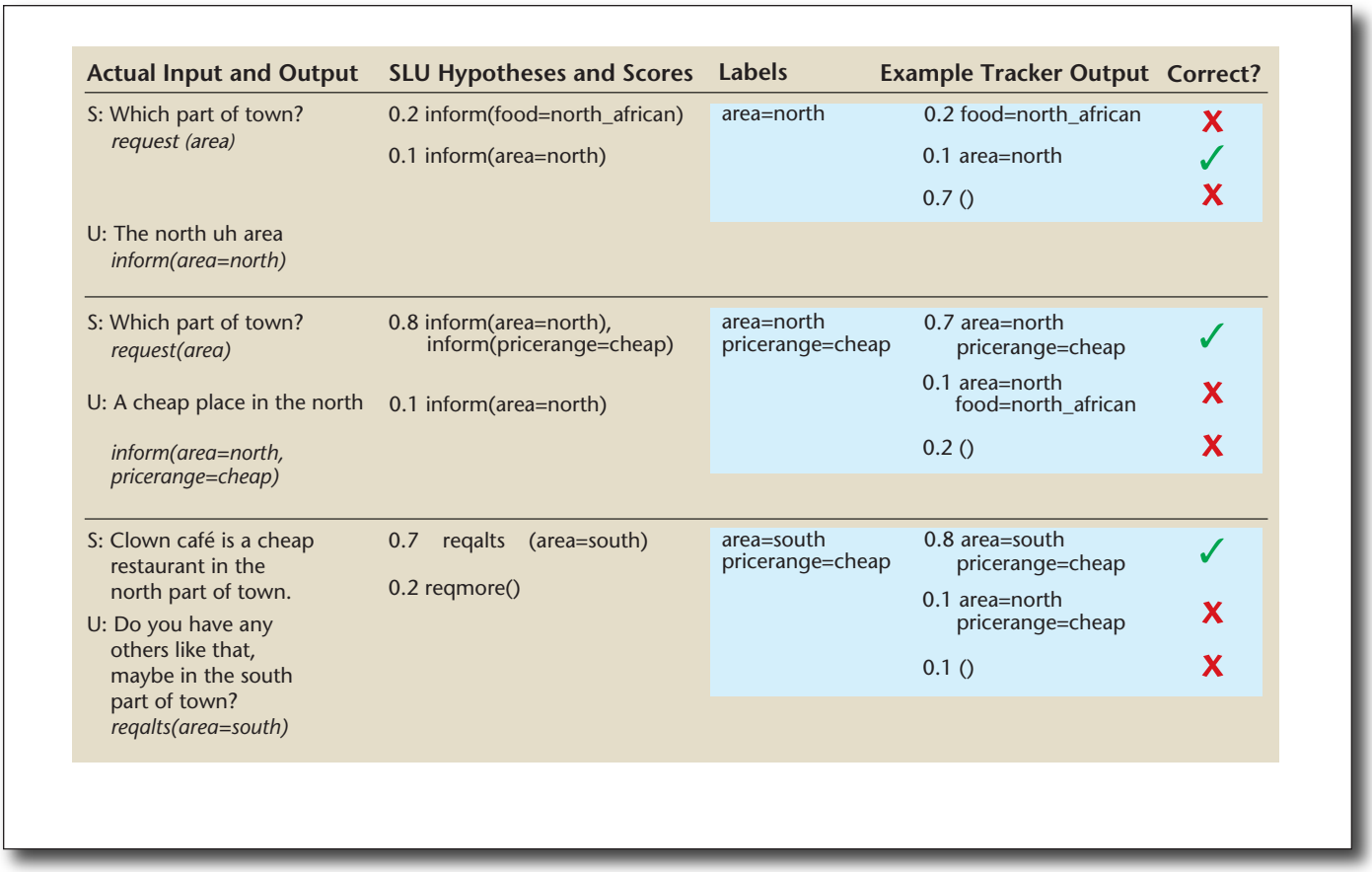


Figure 1. The Dialog State Tracking Problem.

The left column shows the actual dialog system output and user input. The second column shows two SLU n-best hypotheses and their scores. The third column shows the label (correct output) for the user's goal. The fourth column shows example tracker output, and the fifth column indicates correctness.

Challenge Design

The dialog state tracking challenge studies this problem as a corpus-based task. When the challenge starts, labeled human-computer dialogs are released to teams, with scripts for running a baseline system and evaluation. Several months later, a test set of unlabeled dialogs is released. Participants run their trackers, and a week later they return tracker output to the organizers for scoring. After scoring, results and test set labels are made public.

The corpus-based design was chosen because it allows different trackers to be evaluated on the same data, and because a corpus-based task has a much lower barrier to entry for research groups than building an end-to-end dialog system. However when a tracker is deployed, it will inevitably alter the performance of the dialog system it is part of relative to any previously collected dialogs. In order to simulate this mismatch at training time and at run time, and to penalize overfitting to known conditions, dialogs in the test set are conducted using a different dialog manager, not found in the training data.

The first DSTC used 15,000 dialogs between real Pittsburgh bus passengers and a variety of dialog systems, provided by the Dialog Research Center at Carnegie Mellon University (Black et al. 2010). The second and third DSTCs used in total 5,510 dialogs between paid Amazon Mechanical Turkers, who were asked to call a tourist information dialog system and find restaurants that matched particular constraints, provided by the Cambridge University Dialogue Systems Group (Jurcicek, Thomson, and Young 2011).

Each DSTC added new dimensions of study. In the first DSTC, the user's goal was almost always fixed throughout the dialog. In the second DSTC, the user's goal changed in about 40 percent of dialogs. And the third DSTC further tested the ability of trackers to generalize to new domains by including entity types in the test data that were not included in the training data — for example, the training data included only restaurants, but the test data also included bars and coffee shops.

In this relatively new research area, there does not exist a single, generally agreed on evaluation metric; therefore, each DSTC reported a bank of metrics,

sourced from the advisory board and participants. This resulted in approximately 10 different metrics, including accuracy, receiver operating characteristic (ROC) measurements, probability calibration, and so on. Each metric was measured on various subtasks (such as accuracy of a particular component of the user's goal), and at different time resolutions (for example, every dialog turn, just at the end, and so on.) Every combination of these variables was measured and reported, resulting in more than 1000 measurements for each entry. The measurements themselves form a part of the research contribution: after the first DSTC, a correlation analysis was done to determine a small set of roughly orthogonal metrics, which were then reported as featured metrics in DSTC2 and DSTC3, focusing teams' efforts. These featured metrics were accuracy, probability quality (Brier score), and a measure of discrimination computed from an ROC curve.

Each DSTC has been organized by an ad hoc committee, including members of the group providing the dialog data.

Participation and Results

About nine teams have participated in each DSTC, with global representation of the top research centers for spoken dialog systems. Participants have mostly been academic instutions, with a minority of corporate research labs. Results have been presented at special sessions: DSTC1 at the annual Special Interest Group on Discourse and Dialogue (SIGdial) conference in 2013 (Williams et al. 2013); DSTC2 at SIGdial in June 2014 (Henderson, Thomson, and Williams 2014); and DSTC3 at IEEE Spoken Language Technologies (SLT) Workshop in December 2014 (forthcoming).

Papers describing DSTC entries have broken new ground in dialog state tracking; the best-performing entries have been based on conditional random fields (Lee and Eskenazi 2013), recurrent neural networks (Henderson, Thomson, and Young 2014), and web-style ranking (Williams 2014). At present, dialog state trackers are able to reliably exceed the performance of a carefully tuned hand-crafted tracker — for example, in DSTC2, the best trackers achieved approximately 78 percent accuracy versus the baseline's 72 percent. This is impressive considering the maximum performance possible with the provided SLU is 85 percent, due to speech recognition errors.

Prior to the DSTC series, most work on dialog state tracking was based on generative models; however, the most successful DSTC entries have been discriminatively trained models, and these are now the dominant approach. Thus the DSTC series has had a clear impact on the field.

Future Activities

All of the DSTC data will remain available for download, including labels, output from all entries, and the raw tracker output.[1,2] We encourage researchers to use this data for research into dialog state tracking or for other novel uses. In addition, a special issue to the journal *Dialogue and Discourse* will feature work on the DSTC data, and we anticipate publication in 2015. In future challenges, it would be interesting to study aspects of dialog state beyond the user's goal — for example, the user's attitude and expectation. It would also be interesting to consider turn-taking and state tracking of incremental dialogs, where updates are made as each word is recognized. Finally, researchers with dialog data available who would be interested in organizing a future DSTC are encouraged to contact the authors.

Acknowledgements

For DSTC1, we thank the Dialog Research Center at Carnegie Mellon University for providing data, and Microsoft and Honda Research Institute for sponsorship. For DSTC2 and DSTC3, we thank Cambridge University's dialog systems group for providing data. We also thank our advisory committee, including Daniel Boies, Paul Crook, Maxine Eskenazi, Milica Gasic, Dilek Hakkani-Tur, Helen Hastie, Kee-Eung Kim, Ian Lane, Sungjin Lee, Oliver Lemon, Teruhisa Misu, Olivier Pietquin, Joelle Pineau, Brian Strope, David Traum, Steve Young, and Luke Zettlemoyer. Thanks also to Nigel Ward for helpful comments.

Notes

1. research.microsoft.com/events/dstc
2. camdial.org/~mh521/dstc

References

Black, A.; Burger, S.; Langner, B.; Parent, G.; and Eskenazi, M. 2010. Spoken Dialog Challenge 2010. In *Proceedings of the 2010 IEEE Spoken Language Technology Workshop*. Piscataway, NJ: Institute for Electrical and Electronics Engineers.

Henderson, M.; Thomson, B.; and Williams, J. D. 2014. The Second Dialog State Tracking Challenge. In P*roceedings of the 15th Annual SIGdial Meeting on Discourse and Dialogue*. Stroudsburg PA: Association for Computational Linguistics.

Henderson, M.; Thomson, B.; and Young, S. 2014. Word-Based Dialog State Tracking with Recurrent Neural Networks. In *Proceedings of the 15th Annual SIGdial Meeting on Discourse and Dialogue*. Stroudsburg PA: Association for Computational Linguistics.

Jurcicek, F.; Thomson, B.; and Young, S. 2011. Natural Actor and Belief Critic: Reinforcement Algorithm for Learning Parameters of Dialogue Systems Modelled as POMDPs. *ACM Transactions on Speech and Language Processing* 7(3).

Lee, S., and Eskenazi, M. 2013. Recipe for Building Robust Spoken Dialog State Trackers: Dialog State Tracking Challenge System Description. In *Proceedings of the 14th Annual SIGdial Meeting on Discourse and Dialogue*. Stroudsburg PA: Association for Computational Linguistics.

Williams, J. D. 2014. Web-Style Ranking and SLU Combination for Dialog State Tracking. In *Proceedings of the 15th Annual SIGdial Meeting on Discourse and Dialogue*. Stroudsburg PA: Association for Computational Linguistics.

Williams, J. D.; Raux, A.; Ramachadran, D.; and Black, A. 2013. The Dialog State Tracking Challenge. In *Proceedings of the 14th Annual SIGdial Meeting on Discourse and Dialogue*. Stroudsburg PA: Association for Computational Linguistics.

Jason Williams is a researcher at Microsoft Research. His interests include spoken dialog systems, planning under uncertainty, spoken language understanding, and speech recognition. He has published more than 50 journal, conference, and workshop papers and filed more than 10 patents. He is currently vice president of SIGdial — the Special Interest Group on Dialog and Discourse — and served as program cochair for the SIGdial conference in 2013. In the past he has served on the board of directors of the Association for Voice Interaction Design (AVIxD), and on the IEEE Speech and Language Technical Committee (SLTC) in the area of spoken dialog systems. He holds a Ph.D. and master's in speech and language processing from Cambridge University (UK) and a BSE in electrical engineering from Princeton University (USA). Prior to Microsoft, Williams was a principal member of the technical staff at AT&T Labs — Research from 2006–2012.

Matthew Henderson is his Ph.D. under Steve Young at the dialog systems group in Cambridge, UK. His work has looked at statistical methods for dialog systems, particularly in spoken language understanding and dialog state tracking. He studied mathematics at Cambridge for his undergraduate degree, and has a master's degree in speech and language technology from Edinburgh University. He also has a Google Research doctoral fellowship in speech technology.

Antoine Raux is a principal researcher in speech and fusion at Lenovo Inc. His work focuses on various aspects of spoken dialog systems and multimodal human-machine interaction. From 2009 to 2013, Raux was a senior scientist at Honda Research Institute USA, where he led research on human-robot interaction, dialog state tracking, and in-vehicle situated dialog. He is also the main author of the Carnegie Mellon University's Let's Go spoken dialog system, a telephone-based bus schedule information system for the city of Pittsburgh, which has answered more than 200,000 calls from the general public and has been used as a benchmark and a source of real user data throughout the dialog research community. Raux has authored more than 30 papers in peer-reviewed journals and conferences. He holds a Ph.D. in language technologies from Carnegie Mellon University, a master's in intelligence science and technology from Kyoto University (Japan), and an engineering degree from Ecole Polytechnique (France).

Blaise Thomson is chief executive officer and a cofounder of VocalIQ. His interests include robust dialog system design and spoken language understanding and generation. He has published around 40 journal, conference, and workshop papers in the area and completed his Ph.D. and master's degrees in spoken language processing at Cambridge University. Prior to VocalIQ, Thomson was a research fellow at St John's College, Cambridge.

Alan W Black is a professor in the Language Technologies Institute at Carnegie Mellon University (CMU). He did his master's and doctorate at the University of Edinburgh. Before joining the faculty at CMU in 1999, he worked in the Centre for Speech Technology Research at the University of Edinburgh, and before that at ATR in Japan. He is one of the principal authors of the free software Festival Speech Synthesis System and FestVox voice building tools, which constitute the basis for many research and commerical systems around the world. He also works in spoken dialog systems, the Let's Go Bus Information project and mobile speech-to-speech translation systems. Black is an elected member of ISCA board (2007–2015). He has more than 200 refereed publications and is one of the highest cited authors in his field.

Deepak Ramachandran is a research scientist at Nuance Communications Inc. His research interests include dialog management, reinforcement learning, and knowledge representation. He holds a Ph.D. in computer science from the University of Illinois at Urbana-Champaign, and he was previously a scientist at the Honda Research Institute, USA.

Educational Advances in Artificial Intelligence

■ *The Educational Advances in Artificial Intelligence column discusses and shares innovative educational approaches that teach or leverage AI and its many subfields at all levels of education (K-12, undergraduate, and graduate levels).*

Marie desJardins

ACTIVE-ating Artificial Intelligence: Integrating Active Learning in an Introductory Course

■ *This column describes my experience with using a new classroom space (the ACTIVE Center), which was designed to facilitate group-based active learning and problem solving, to teach an introductory artificial intelligence course. By restructuring the course into a format that was roughly half lecture and half small-group problem solving, I was able to significantly increase student engagement, their understanding and retention of difficult concepts, and my own enjoyment in teaching the class.*

In spring 2013, several colleagues and I received an award, from the Hrabowski Fund for Innovation[1] at the University of Maryland Baltimore County, to create a new classroom, using additional funding donated by BAE Systems and Northrup Grumman. The ACTIVE Center[2] is designed to provide a dynamic physical and virtual environment that supports active, collaborative learning; skill mastery through in-class problem solving; and laptop-based in-class laboratory activities. The ACTIVE Center's design was based on research on the power of collaborative learning to promote student success and retention, particularly for women, underrepresented minorities, and transfer students, who benefit greatly from building stronger connections with their peers through shared active learning experiences (Zhao, Carini, and Kuh 2006; Rypisi, Malcolm, and Kim 2009; Kahveci, Southerland, and Gilmer 2006).

The ACTIVE Center, a 40-student classroom, includes movable furniture (20 trapezoidal tables and 40 lightweight rolling chairs) that is typically grouped into 10 hexagonal table clusters but that can also be arranged into lecture-style rows, a boardroom or seminar-style rectangular layout, or individual pair-activity tables. The room also has an Epson Brightlink "smart projector" at the front of the room, four flat-panel displays (which can be driven centrally by the instructor's laptop or individually through HDMI ports), and 10 rolling 4 x 6 foot whiteboards for use during group problem-solving activities, as well as smaller, portable tabletop whiteboards. The ACTIVE Center was ready for use in early February 2014, and we moved several classes from regular classrooms into the new space, including my undergraduate introduction to AI (CMSC 471).

Over the last 12 years of teaching introductory AI, I had gradually moved toward incorporating more problems and

 ISSN 0738-4602

exercises into my lecture slides and making the class very interactive. However, I was never completely successful at convincing students to work independently on problem solving during the class —– many students would get stuck or distracted, and it was difficult to diagnose their level of understanding. This semester, with a physical environment that was designed to facilitate in-class problem solving, I decided to take full advantage of it, setting a goal of a roughly equal mix of lecture and problem solving.

I created a prereading assignment for each class day that included a short introduction to the basic concepts. I often led off the lecture part of the class with a *mini-quiz* (a slide with questions that students ought to know the answers to from the reading) and a very quick recap of those basic concepts. (Previously, I had never successfully convinced students to do the textbook reading of Russell and Norvig's *Artificial Intelligence: A Modern Approach* [2010] before class. Now, most students, but not all, did this preading.) I'd then dive into the more advanced material that wasn't covered in the prereading, spending around half of the class on lecture and board-based problem solving (taking advantage of the smart projector to do screen captures of group solutions), and the rest of the 75-minute class period having small groups of four or five students working on more challenging problems.

During problem-solving sessions, students would bring one of the wheeled whiteboards to their table and work on an assigned problem. Once the students got used to the format, they didn't need any urging to get started on their work. I didn't assign the groups (they evolved naturally based on where the students chose to sit) or roles (some groups rotated roles, and in others, it was almost always the same person at the whiteboard). But as I circulated, it was obvious that every single student in the class was engaged with the process — paying attention, contributing, and thinking. It was actually quite remarkable — in a class of 40 students, there was literally not one single person who wasn't involved in problem solving during those parts of the class. Moreover, I could tell which groups understood the concepts and were making progress and which groups weren't. I could work individually with groups who were stuck, and I could identify errors that multiple groups were making, bringing the class's attention back to talk about those misconceptions with the whole class. It was an extremely effective way to mix coaching, remediation, and discussion.

The format did vary somewhat, including days where lectures predominated; where lectures and problem solving were interspersed; or "Lisp labs," where students used their laptops to work on Lisp coding with some instructor guidance. We also rearranged the room into a seminar style layout for a class debate on Searle's "Minds, Brains, and Programs" (1980) and Raymond Kurzweil's theories about the singularity.

I collected assessment data through student and instructor surveys (in all classes offered in the ACTIVE Center), but have not yet systematically analyzed the data. I did not see a significant difference in exam grades or overall course grades compared to my 2011 offering, but my anecdotal observation is that the students did better on the problem-solving parts of the exam but less well on the "details of advanced methods" questions. That makes sense: we spent more time on problem solving and less time covering details, and I don't think that students "filled in the gaps" by spending more time on the reading. How to get both deep conceptual learning and broad understanding of different types of methods and techniques is a continual goal for reflection. Some of the other challenges and ideas for the future include managing class pacing when alternating between lecture and problem solving, designing problems of appropriate difficulty, and creating in-class laptop-based activities to explore AI concepts at the implementation level.

All of my course materials (syllabus, schedule, reading and prereading assignments, PowerPoint slides, which include whole-class and group problem-solving activities, and homework assignments) are posted on the course website.[3] Colleagues are welcome to reuse these materials with attribution; I would greatly appreciate any feedback or experience reports that you would be willing to share. Having taught introductory AI eight times, I can say with confidence that despite feeling some pressure about whether the problem-solving format would work well, this semester was the most fun that I've had teaching an AI course. It would be very hard to return to a regular classroom and to a standard lecture-based presentation style. I strongly encourage other institutions to consider creating or retrofitting classrooms using design practices that would facilitate this kind of coursework and learning environment.

Notes

1. See innovationfund.umbc.edu.
2. See active.umbc.edu.
3. See www.csee.umbc.edu/courses/undergraduate/CMSC471/spring14.

References

Kahveci, A.; Southerland, S. A.; and Gilmer, P. J. 2006. Retaining Undergraduate Women in Science, Mathematics, and Engineering. *Journal of College Science Teaching* 36(3): 34–38.

Kurzweil, Ray 2005. *The Singularity Is Near: When Humans Transcend Biology*. New York: Viking.

Russell, S., and Norvig, P. 2010. *Artificial Intelligence: A Modern Approach,* 3rd ed. Englewood Cliffs, NJ: Prentice Hall.

Rypisi, C.; Malcolm, L. E.; and Kim, H. S. 2009. Environmental and Developmental Approaches to Supporting Women's Success in STEM Fields. In Student Engagement in Higher Education, ed. S. R. Harper and S. J. Quay, 117–135. New York: Routledge.

Searle, J. R. 1980. Minds, Brains, and Programs. *Behavioral and Brain Sciences* 3(3)(September): 417–424.

Zhao, C. M.; Carini, R. M.; and Kuh, G. D. 2006. Searching for the Peach Blossom Shangri-La: Student Engagement of Men and Women SMET Majors. *Review of Higher Education* 28(4): 503–25.

Marie desJardins is a professor of computer science and electrical engineering at the University of Maryland, Baltimore County. Her research is in artificial intelligence, with current interests in planning, learning, and multiagent systems.

AAAI News

Winter News from the Association for the Advancement of Artificial Intelligence

AAAI-15 and IAAI-15 Are Almost Here!

The Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15) and the Twenty-Seventh Conference on Innovative Applications of Artificial Intelligence (IAAI-15) will be held January 25–30 at the Hyatt Regency Austin in Austin, Texas, USA. Austin is home to one of the largest and oldest communities of AI researchers in the world, with more than a dozen AI-related labs at the University of Texas at Austin.

Included here are a few highlights, but for the full program, links, and schedule information, please visit aaai.org/aaai15.

Robotics!

AAAI will feature a host of robotics exhibitions, demonstrations, and invited talks. In addition to the AI Robotics Early Career Spotlight Talk and Robotics in Texas program, a few highlights of the Robotics Festival will include the following:

50 Years of AI & Robotics

Please join us for this special panel celebrating the 50th anniversary of Shakey, which will be held on Tuesday, January 27, cosponsored by AAAI and the IEEE Robotics and Automation Society, the IEEE International Conference on Robotics and Automation and the Robotics: Science and Systems Conference. Panelists will include Edward Feigenbaum, Peter Hart, and Nils Nilsson.

RoboCup Soccer Exhibition

Given the recent expansion of interest in intelligent robotics, AAAI and the RoboCup Federation, with the help of NSF, are cosponsoring a RoboCup soccer exhibition match at AAAI-15 (January 26–27) to showcase the state-of-the-art in robotics soccer to the broad artificial intelligence research community and spur additional interest in this exciting testbed for intelligent systems.

The RoboCup competitions have promoted research on artificial intelligence and robotics since 1997. One of their main foci is the worldwide popular game of soccer, with the aim to build fully autonomous cooperative multi-robot systems that perform well in dynamic and adversarial environments.

The participating teams in the AAAI-15 event will be UPennalizers from The University of Pennsylvania, UT Austin Villa from the University of Texas at Austin, and rUNSWift from the University of New South Wales. Each team won a championship at the 2014 international competition (in the humanoid league, 3D simulation league, and Standard Platform League respectively). They will demonstrate a game according to the regulations of the Standard Platform League, in which all teams use identical Aldebaran Nao robots.

Robotics Exhibition

Research groups and robotics companies will participate in the AAAI Robotics Exhibition, demonstrating their robot systems throughout AAAI-15. The Exhibition will span the AAAI-15 Open House program on Monday, January 26 through Thursday, January 29. The open house will be open to high-school students and other selected members of the general public. Demonstrations that highlight progress in robotics during the past five years will be featured.

NSF-Sponsored Workshop on Research Issues at the Boundary of AI and Robotics

This full-day workshop on January 25 is an initiative resulting from cooperation between AAAI and the IEEE. It is designed to bring together AI experts, robotics experts and program directors to compile a list of recommendations to funding agencies, professional organizations and individual researchers for how to push the boundary of AI and robotics.

Students!

As part of AAAI's outreach to students in 2015, AAAI-15 will include many special programs specifically for students, including the following:

Student Newcomer Lunch: AAAI will holds its inagural Student Newcomer Lunch on Sunday, January 25, at 1:00 PM. This lunch is designed to welcome first-time students to the conference and give them an opportunity to meet other participants before the launch of the main conference.

AAAI-15 Open House: The AAAI-15 Open House will be held Monday, January 26, for high-school students in the Austin area, the general public, graduate and undergraduate students, and established AI researchers. The latest work in many areas of AI will be showcased.

Breakfast with Champions: A Women's Mentoring Event: AAAI is holding an inaugural women's mentoring event between women students and senior women in CS/AI. Breakfast with Champions will be Wednesday morning, January 28 at 7:15 AM. This event follows on the successful footsteps of the Doctoral Consortium Mentoring program.

Information about these and all student programs is available on the AAAI Registration form at www.regonline.com/aaai15 and at http://movingai.com/AAAI15/.

ICWSM-15

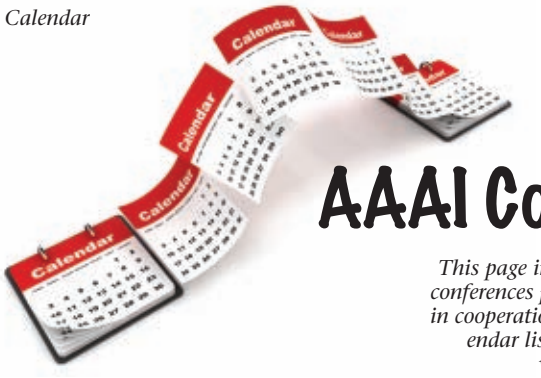
The Ninth International AAAI Conference on Web and Social Media (ICWSM) will be held May 26-29, 2015 in Oxford, England, UK.

January 18: Abstracts Due
January 23: Papers, Posters, and Demos Due. See www.icwsm.org for details.

2015 AAAI Spring Symposia

The Twenty-Eighth AAAI Spring Symposium Series will be held March 23–25 at Stanford University, in cooperation with the Stanford Computer Science Department. The eight symposia are (1) Ambient Intelligence for Health and Cognitive Enhancement; (2) Applied Computational Game Theory; (3) Foundations of Autonomy and Its (Cyber) Threats: From Individuals to Interdependence; (4) Knowledge Representation and Reasoning: Integrating Symbolic and Neural Approaches; (5) Logical Formalizations of Commonsense Reasoning; (6) Socio-Technical Behavior Mining: From Data to Decisions; (7) Structured Data for Humanitarian Technologies: Perfect Fit or Overkill? and (8) Turn-Taking and Coordination in Human-Machine Interaction.

February 27, 2015 is the general registration deadline. The website is located at www.aaai.org/Symposia/Spring/



AAAI Conferences Calendar

This page includes forthcoming AAAI sponsored conferences, conferences presented by AAAI Affiliates, and conferences held in cooperation with AAAI. AI Magazine *also maintains a calendar listing that includes nonaffiliated conferences at www.aaai.org/Magazine/calendar.php.*

AAAI Sponsored Conferences

Twenty-Ninth AAAI Conference on Artificial Intelligence. AAAI-15 will be held January 25–30 in Austin, Texas, USA.

URL: www.aaai.org/aaai15

AAAI Spring Symposium. The AAAI Spring Symposium Series will be held March 23–25 in Palo Alto, CA USA.

URL: www.aaai.org/Symposia/Spring/sss15.php

The 9th International AAAI Conference on Weblogs and Social Media. ICWSM 2015 will be held May 26–29 in Oxford, UK.

URL: www.icwsm.org/2015

Conferences Held by AAAI Affiliates

Twenty-Eighth International Florida AI Research Society Conference. FLAIRS-15 will be held May 18–20, 2015 in Hollywood, Florida, USA

URL: www.flairs-28.info

Twenty-Fifth International Conference on Automated Planning and Scheduling. ICAPS-15 will be held June 7–11, 2015 in Jerusalem, Israel

URL: icaps15.icaps-conference.org

International Joint Conference on Artificial Intelligence. IJCAI-15 will be held July 25 – August 1, 2015 in Buenos Aires, Argentina.

URL: ijcai-15.org

Conferences Held in Cooperation with AAAI

10th ACM/IEEE International Conference on Human-Robot Interaction. HRI 2015 will be held March 1–4 in Portland, Oregon USA

URL: humanrobotinteraction.org/2015

7th International Conference on Agents and Artificial Intelligence. ICAART 2014 will be held January 10–12 in Lisbon, Portugal

URL: www.icaart.org

8th International Joint Conference on Biomedical Engineering Systems and Technologies. BIOSTEC 2015 will be held January 12–15 in Lisbon, Portugal

URL: www.biostec.org

4th International Conference on Pattern Recognition Applications and Methods. ICPRAM 2015 will be held January 10–12 in Lisbon, Portugal

URL: www.icpram.org

17th International Conference on Enterprise Information Systems. ICEIS 2015 will be held April 27–30, 2015, in Barcelona, Spain

URL: www.iceis.org

15th International Conference on Artificial Intelligence and Law. ICAIL 2015 will be held June 8–12, in San Diego, California, USA

URL: sites.sandiego.edu/icail

28th International Workshop on Qualitative Reasoning. QR 2015 will be held August 10–12, in Minneapolis, MN, USA

URL: qr15.sift.net

 ISSN 0738-4602

AAAI-15 Austin, Texas USA
The First *Winter* AI Conference!

The Twenty-Ninth AAAI Conference on Artificial Intelligence

The Twenty-Seventh Conference on Innovative Applications of Artificial Intelligence

January 25–30 2015

Austin, Texas USA

www.aaai.org/aaai15

www.aaai.org/iaai15