6-2-2014

The Use of the Blackboard Architecture for a Decision Making System for the Control of Craft with Various Actuator and Movement Capabilities

Jeremy Straub

Hassan Reza
University of North Dakota, hassan.reza@UND.edu

The Use of the Blackboard Architecture for a Decision Making System for the Control of Craft with Various Actuator and Movement Capabilities

Jeremy Straub; Hassan Reza

Abstract:

This paper provides an overview of an approach to the control of multiple craft with heterogeneous movement and actuation characteristics that is based on the Blackboard software architecture. An overview of the Blackboard architecture is provided. Then, the operational and mission requirements that dictate the need for autonomous control are characterized and the utility of the Blackboard architecture is for meeting these requirements is discussed. The performance of a best-path solver and naïve solver are compared. The results demonstrate that the best-path solver outperforms the naïve solver in the amount of time taken to generate a solution, however, the number of solver-runs to be executed against the Blackboard must be sufficient to allow the lower individual-run times to offset the time required to propagate the data utilized by the best-path solver for solution generation through the database. The existence of other justifications for this approach (even if the number of runs for each data propagation cycle is not sufficient) is also discussed.

I. Introduction

The utility of robots for a wide variety of applications has been demonstrated. These applications include assisting humans in human-present situations, operating under human remote control (teleoperation) and performing tasks in environments where direct human control is not possible or is undesirable. This last category includes applications where human control would exceed communications capabilities (e.g., a swarm of robots deployed for a reconnaissance application) or where real-time human control is not possible due to communications latency (e.g., planetary exploration).

In circumstances where robots must operate on their own without direct real-time teleoperation, performance can be increased by providing them with goals instead of commands. Carsten, et al. [1], forexample, demonstrate how the performance of Martian rovers was enhanced by the use of more robust, movement goal-driven planning and routing software. This can be further enhanced by allowing a robot to autonomously generate component goals, based on a higher-level goal assigned to it (e.g., [2]–[3][4]).

The Blackboard architecture has been proposed for the control of robotic systems. It works by evaluating rules, which are triggered by the presence of requisite facts; rules can trigger actions and/or assert facts based on their analysis. While a Blackboard system can operate in this mode (taking whatever information exists or is supplied to it and triggering actions based on this), evaluating potential paths to a desired goal or goals can allow improved performance via targeting resources at the areas that are projected to provide the greatest benefit (in terms of movement towards a solution).

This paper compares two approaches for identifying routes on the Blackboard: a naïve approach which simulates operations in the forward-only mode (but does this based on projectionsof what data will be collected) and a best-path solver which propagates costs throughout the Blackboard and then finds the best-available (lowest cost) rote from current conditions to the specified destination.

II. Blackboard

The work discussed herein draws from prior research in two related areas. Prior work related to the Blackboard architecture and autonomous craft control will now be reviewed.

A. Blackboard Architecture

The Blackboard architectural style has been employed in applications that demand complex interpretation of knowledge bases shared by a set of independent programs known as knowledge sources (or agents) [5]. The essential components of this architecture include: the Blackboard controller and a set of knowledge sources, which interact with one another using the blackboard.

The Blackboard architecture is an extension of the concept of an expert system [6]–[7][8], instead of making recommendations, however, it can trigger actions and influence its operating environment. Hayes-Roth's [9] Blackboard architecture concept (based on previous work on the Hearsay-II system [10]) included two blackboards: one serviced control operations while the other dealt with the subject matter (e.g., the exploration goals). Numerous other configurations exist, however. The blackboard includes rules, actions and facts. Rules are deemed invokable if their preconditions (facts that must be true, or within a value range in some implementations) are met and rules are selected from the invokable list for execution. A rule that executes can assert facts or invoke actions. Actions perform a task and may invoke other actions and/or assert facts.

The Blackboard architecture has been used in a variety of applications. In addition to Hayes-Roth's work, these include robotic control [11]–[12][13][14][15], e-learning [16], [17], network management [18], application testing [19], speech recognition [20], disaster response [21], software selection [22], caring for the elderly [23], project scheduling [24], risk management [25], protein modeling [26], [27], system problem diagnosis [28], [29], military control [30] and even preventing terrorism [31].

The original blackboard concept has been expanded in a variety of ways. For example, Rice [32] and Hewett and Hewett [33] implemented languages for implementing Blackboard systems. Le Mentec and Brunessaux worked on real-time response capabilities [34].

B. Autonomous Control of Multiple Robot Systems

A variety of multi-robot systems have been proposed. Sensornets [35-37], for example, may include robots (inclusive of ground, aerial and orbital craft) among their sensor collection. Fink [38-40] proposed a centrally controlled “tierscalable” robotic system. Prior work [41-43] has discussed the utility of a distributed approach to this control. Corke, Peterson and Rus [44] have proposed a two-tier network of camera-guided UAVs, demonstrating a potential problem related to communications which is characterized in other work by Boano, et al. [45]. Fairbairn, Bate and Stankovic [46] proffer that environmental factors pose the largest threat to this type of

system. Solutions to communications-related problems are proposed by De Poorter, Moerman and Demeester [35] and Tate, Bate and Poulding [47]. Data transmission limitations are also an important consideration, solutions to this include approximation [48] and increasing the value-perunit of the data that is transmitted [49-51]. Training solutions have been proposed by Colby and Tumer [52], Zhang and Lesser [53], Hamzi, et al. [54], Chun, et al. [55] and Haghighi [56].

III. Multi-Tier Blackboard Control System

A Blackboard-based control system for a hierarchy of craft with heterogeneous movement and actuation characteristics was proposed in [43]. This system functions by defining a missionin terms of various high-level goals and a set of rules, facts and actions that allow the craft to find pathways to achieving these goals. Final conditions (initially designed to be rules, but later switched to facts to allow actions to directly trigger a final condition instead of having to wait for the applicable rule to be selected from the invokable list and some simplification benefits) are defined by system users. The system can have multiple final conditions, the satisfaction of any of which is deemed to indicate goal completion (if multiple conditions are required then a rule that accepts these as a pre-condition can be created). A collection of craft is delegated an overall mission goal (or submission goal) by human controllers. Goals, based on an autonomously-defined work breakdown structure, are then delegated within the collection of craft to lower-level group managers and to craft that will carry out the work.

Once a goal is defined, the blackboard solver goes to work and identifies a candidate path to goal completion. This path is refined as additional information is known (e.g., from data collection activities). This path is then used by the decision-making system. Figure 1presents a diagram of the system's operations. The controller supplies high level goals which are translated into final conditions. With these conditions identified, system iteration begins. During each iteration, the system

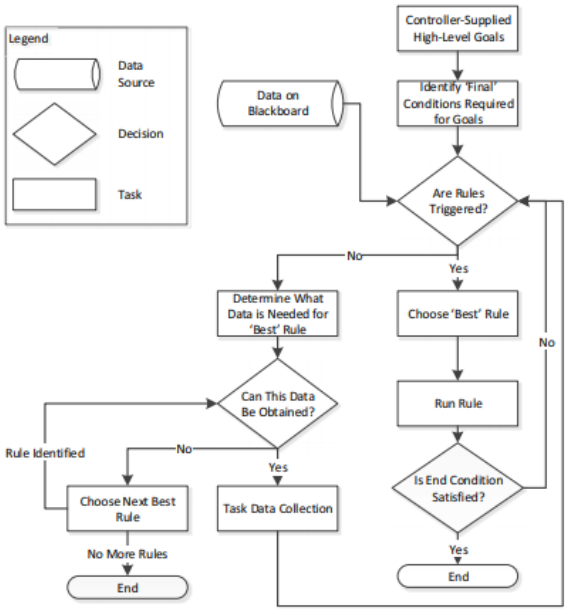

Figure 1. Diagram of System Operations (modified from [43])

checks to see if rules are triggered. Based on the presumption that rules are less expensive than actions (which generally involve activities in the real world), triggered rules are executed

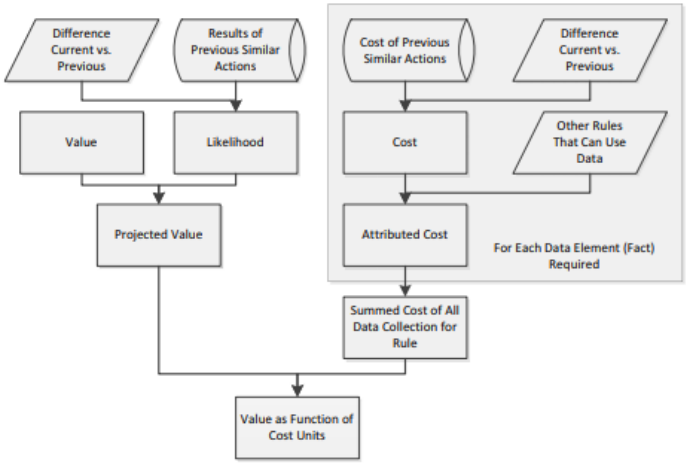

Figure 2. Score Determination for Each Rule [43].

before identifying untriggered rules to attempt to satisfy the preconditions for. Rules are selected (either from the triggered but unexecuted list or to attempt to procure data to satisfy) based on their usefulness in reaching a final condition. After each rule is run, the system checks whether a final condition exists; if not, it restarts (looking first for triggered, unexecuted rules and, then, for the best un-triggered rule).

When processing un-triggered rules, the system attempts to determine what rule will move the system furthest towards a final condition relative to its cost. One the best rule is selected, its preconditions are assessed for plausibility. If they are not plausible, the system looks at the next best rule (this continues until a rule with plausible-to-collect data requirements is located or no more rules exist). Once a plausible rule is found, the data collection is tasked and the system return to checking for triggered rules (the collected data will be inserted onto the blackboard when it is available, potentially triggering rules). Depending on configuration, the system may identify additional data collection tasks to queue or it may be prohibited from considering additional data collection until the data collection under way is complete.

The process of determining the value (a combination of cost and benefit) of each prospective rule to trigger is depicted in Figure 2. The process begins with computing the projected value which is based on the value of the data and the likelihood of the projected data element being collected (based on the difference between the current data and previously collected data and the results of similar data collection tasks). The attributed cost is also determined. The cost is determined (based on the cost of previous actions and the similarity of the current action to previous ones). The attributed cost is determined by the cost and the amount of the cost that is attributed to the particular rule. The attributed cost for each required fact is summed. Finally, the value is determined from these two numbers (including coefficients that allow weighting between the cost and benefit).

IV. Blackboard Solver Implementation

The two different types of blackboard solvers that are compared in this work are now briefly discussed. First, the naïve solver is presented; then, the best-path solver is discussed.

A. Naïve Solver

The naïve solver works by simulating the operations of the blackboard's rules and actions. Invokable rules are selected (based on their order) and executed; these rules trigger actions (which, themselves may trigger other actions or assert facts) and assert facts. The asserted facts may satisfy the pre-conditions of other rules, which are then placed into the invokable list for later selection for execution.

For the purposes of path generation, it is presumed that the actions will generate the data that they are projected to generate (based on known characteristics of the data collection area, the performance of past actions of a similar type and the similarity between the current action and similar past actions). The incorporation of a confidence metric and its effect on path generation is a subject for future work.

B. Best-Path Solver

The best path solver begins by propagating cost values through the rule / action / fact network. It performs this iteratively, replacing costs assigned to given nodes if a lowercost route to the node is found. This propagation process continues until an iteration is completed with no changes being made to the node network.

Once these values are propagated, the solver identifies the lowest-cost route from asserted facts via currently invokable rules and the actions that they trigger to the desired end condition. A recursive search algorithm is utilized to perform this path finding process for the network of nodes.

V. Experimental Data

This section provides an overview of the experimental methodology, the simulation software that was used for data collection, and the data that was collected. This data is analyzed in the subsequent section.

A. Methodology

The blackboard implementation and solvers, as described in Sections III and IV, where implemented. This application also has the capability to create random networks of facts, rules and actions for testing. For the purposes of the tests performed herein, a network consisting of 1,000 facts, 1,000 rules and 1,000 actions was created. Approximately 30% of the facts were initially asserted. For each rule a random selection of the number of pre-conditions (between 1 and 7) and triggered items (between 0 and 7) was made. The pre-condition facts and triggered facts or actions (as well as whether a fact or action was chosen) were selected randomly as well. Actions were randomly assigned a cost between 0 and 25.

Ten data collection runs were performed. Each run began with the generation of a new network and the random selection of a new target node (which served as a single final condition for the experiment). During each run, both the best path and naïve solvers were run on the network.

B. Simulation Software

A simulation program was created to perform the work described in the preceding subsection. This software collected the data which is presented in the following subsection; it also acted as the environment that the blackboard system operated within, supplying responses to the actions that were executed. To avoid confounding the desired analysis, actions were deemed to always assert the facts that they were projected to assert (even though, in reality, this would not always be the case), as a probabilistic model for action results would simply have served to add noise to the data collection and analysis process. As the likelihood of action results is application-specific, the analysis of this will serve as a topic for future work in the context of a particular application.

Each simulation run began with a newly generated blackboard, which provided the data presented for both the naïve and best-path solvers. Manual spot-checking was used to verify that the simulation-generated data was being generated (i.e., the system was functioning properly) and recorded (i.e., the simulation software was measuring the desired elements) correctly.

C. Data Collected

The process presented in the previous subsections was performed and the data collected during this is shown in Table 1. This table lists the run number as well as the number of solver iterations, solution hops and rules and actions run for each test. The solver iterations and solution hops characterize the performance of the best-path solver, while the number of rules and actions run characterize the performance of the naïve solver.

The number of solver iterations describes how many times the network was processed to propagate the aggregate cost values through it. The solver iterates until it completes a run without any changes being made to cost values. The number of solution hops is the number of invocations that are included in the identified best path.

For the naïve solver, the number of rules and actions that are run are recorded. The naïve solver operates until the designated final condition is reached.

TABLE I. COLLECTED DATA

Run	*Solver Iterations*	*Solution Hops*	*Rules Run*	*Actions Run*
1	5	2	406	845
2	14	4	479	954
3	4	4	414	777
4	4	2	2064	3876
5	14	22	438	857
6	3	3	338	654
7	12	32	1785	3455
8	6	3	409	815
9	6	2	374	766
10	4	3	484	1038

For the purposes of visual comparison, this data is also presented as bar graphs. Figure 3 presents the number of solver iterations required. Figure 4 presents the number of hops required for each run and Figure 5 presents the number of rules and actions required. It is notable that there is little evident correlation between the number of iterations of the solver required, the number of elements in the best solution (solution hops) and the number of rules and actions that must be run.

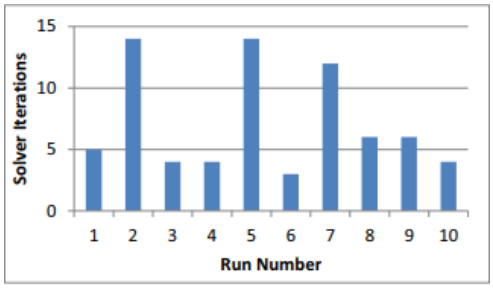

Figure 3. Solver Iterations.

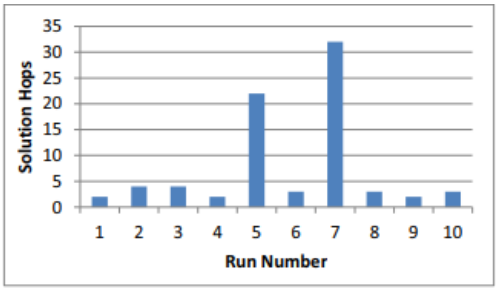

Figure 4. Solution Hops.

In order to assess the foregoing, knowledge of additional details is required. An average amount of computational time per solver iteration (best-path solver) and a characterization of the time required for running rules and actions (naïve solver) is required.

Figure 5. Number of Rules and Actions Run.

Data from ten trials (these are separate from the trials describing Table 1) was used to generate an average time-periteration value. In these tests, an average of 10.1 solver iterations was required which took an average of 26.1 milliseconds. Thus, the solver takes approximately 2.6 milliseconds per iteration.

Rules and actions (from a computational standpoint) take a roughly similar amount of time to run. Ten trials were run to characterize an average amount of time per rule/action. An average of 590 rules and 1173.2 actions were run per trial, requiring an average of 3.5 milliseconds. Thus, an average of 0.002 milliseconds is required for running each rule or action.

VI. DATA ANALYSIS

The data in Section V facilitates an analysis of the comparative performance of the two approaches. The amount of computational time required is presented in Table 2. From this table, it is clear that the naïve approach outperforms the computational solver, in terms of the required computational time. The best path approach requires 4.4 times as long to reach a solution.

TABLE II. COMPUTATIONAL TIME REQUIREMENTS

Run #	*Best Path Solver*	*Naïve Approach*
1	13	2.502
2	36.4	2.866
3	10.4	2.382
4	10.4	11.88
5	36.4	2.59
6	7.8	1.984
7	31.2	10.48
8	15.6	2.448
9	15.6	2.28
10	10.4	3.044
Average	18.72	4.2456

This benefit, however, is dwarfed by the number of actions that are required (on average 1403.7, for the data presented in Table 1, versus below 10 for the best-path approach). While the arbitrary cost assignment (from 0 to 25) has no direct real-world correlation (as this would vary by application and the time and resources required to perform each data collection action), the difference (17,546.25, on average, versus 125 or less) is significant. The best-path solver out performs the naïve approach by two orders of magnitude. The amount of time and resource savings (as well as the practical significance of these savings) must be determined on an application-specific basis. This will serve as a topic for future work.

VII. Conclusions And Future Work

A decision making and control approach based on the Blackboard architecture have been presented. The use of this for the purposes of controlling a collection of craft with heterogeneous movement and actuation capabilities has been discussed. A key component of this system, the solver that is used to determine the path against which rules are valued and costs are assessed to determine what data is tasked for selection has been discussed. Two approaches for the development of this component have been presented and compared. The first, a naïve solver, takes less computational time, but produces inferior paths. The best-path solver, on the other hand, finds an optimal path but takes more than four times the computational time to do so. The exact level of trade-off between these two approaches requires knowledge of the specific tasks being performed; however, as actions generally are significantly more expensive (both in terms of time and cost), the best-path solver is, thus, generally preferable. In the work presented, the naïve solver was only run once per iteration; however, that naïve solver could be run up to (on

average) four times and still have less computational cost than the best-path solver. The comparison of this is a subject for future work.

Future work will also include additional testing on other aspects of the proposed control system and its expansion into a distributed Blackboard-style system, allowing for testing of its ability to locally control the collection of heterogeneous craft. This will also facilitate the comparison of the proposed approach to other non-Blackboard-based approaches.

References

[1] J. Carsten, A. Rankin, D. Ferguson and A. Stentz. Global path planning on board the mars exploration rovers. Presented at Aerospace Conference, 2007 IEEE. 2007.

[2] J. Straub. Model based data transmission: Analysis of link budget requirement reduction. Communications and Network 4(4), pp. 278-287. 2012.

[3] J. Straub, "Multi-tier exploration: An architecture for dramatically increasing mission ROI," in Proceedings of the AIAA Space 2012 Conference, Pasadena, CA, USA, 2012.

[4] J. Straub. Increasing interplanetary CubeSat mission science return with model based transmission reduction. Presented at 1st Interplanetary CubeSat Workshop. 2012, .

[5] F. Buschmann, R. Meunier, H. Rohnert and M. Stal. Pattern-Oriented Software Architecture: A System of Patterns 1996.

[6] D. Waterman. A guide to expert systems. 1986.

[7] B. G. Buchanan, D. Barstow, R. Bechtal, J. Bennett, W. Clancey, C. Kulikowski, T. Mitchell and D. A. Waterman. Constructing an expert system. Building Expert Systems 50pp. 127-167. 1983.

[8] R. M. O"Keefe, O. Balci and E. P. Smith. Validation of expert system performance. 1986.

[9] B. Hayes-Roth. A blackboard architecture for control. Artif. Intell. 26(3), pp. 251-321. 1985.

[10] L. D. Erman, F. Hayes-Roth, V. R. Lesser and D. R. Reddy. The hearsayII speech-understanding system: Integrating knowledge to resolve uncertainty. ACM Computing Surveys (CSUR) 12(2), pp. 213-253. 1980.

[11] C. W. Fox, M. H. Evans, M. J. Pearson and T. J. Prescott. Towardshierarchical blackboard mapping on a whiskered robot. Robotics and Autonomous Systems 60(11), pp. 1356-1366. 2012.

[12] H. Reza, and K. Ogaard. Modelling UAS Swarm System Using Conceptual and Dynamic Architectural Modeling Concepts. Conceptual Structures for Discovering Knowledge. Andrews et al (Eds.), LNAI 6828, Springer -Verlag, 2011.

[13] A. Paraschos, N. I. Spanoudakis and M. G. Lagoudakis. Model-driven behavior specification for robotic teams. Presented at Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1. 2012.

[14] G. Brzykcy, J. Martinek, A. Meissner and P. Skrzypczynski. Multi-agent blackboard architecture for a mobile robot. Presented at Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference On. 2001.

[15] R. E. Fayek, R. Liscano and G. M. Karam. A system architecture for a mobile robot based on activities and a blackboard control unit. Presented at Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference On. 1993.

[16] F. Jurado, M. A. Redondo and M. Ortega. Blackboard architecture to integrate components and agents in heterogeneous distributed eLearning systems: An application for learning to program. J. Syst. Software 85(7), pp. 1621-1636. 2012.

[17] M. Huang, H. Chiang, P. Wu and Y. Hsieh. A multi-strategy machine learning student modeling for intelligent tutoring systems: Based on blackboard approach. Library Hi Tech 31(2), pp. 6-6. 2013.

[18] G. Prem Kumar. Integrated network management using extended blackboard architecture. 2013.

[19] H. Chu. A blackboard-based decision support framework for testing client/server applications. Presented at Software Engineering (WCSE), 2012 Third World Congress On. 2012.

[20] Y. Cao. Polyphonic transcription using blackboard approach: Summary. 2012.

[21] Q. Jiang, H. Y. Hu, X. F. Jia, C. P. Lu and R. J. Wang. A hybrid knowledge-base system approach for disaster emergency and relief decision. Advanced Materials Research 791pp. 2234-2237. 2013.

[22] K. Eldrandaly and S. Naguib. A knowledge-based system for GIS software selection. Int.Arab J.Inf.Technol. 10(2), pp. 152-159. 2013.

[23] P. Wu, H. Peng, J. Zhu and Y. Zhang. "Senscare: Semi-automatic activity summarization system for elderly care," in Mobile Computing, Applications, and Services. 2012.

[24] S. Adhau, M. Mittal and A. Mittal. A multi-agent system for distributed multi-project scheduling: An auction-based negotiation approach. Eng Appl Artif Intell 25(8), pp. 1738-1751. 2012.

[25] E. Y. Sanchez and A. A. Acquesta. CRISIS: A system for risk management. Systems 1(1), pp. 3-26. 2012.

[26] I. Muscalagiu, H. E. Popa, M. Panoiu and V. Negru. "Multi-agent systems applied in the modelling and simulation of the protein folding problem using distributed constraints," in Multiagent System Technologies. 2013.

[27] M. V. Johnson Jr and B. Hayes-Roth. Integrating diverse reasoning methods in the BBP blackboard control Architecture1. Presented at Proceedings of the AAAI. 1987.

[28] J. Ronchi, G. Butera, E. Frascari and P. Scaruffi. A distributed blackboard-based architecture for tele-diagnosis. Artificial Intelligence for Engineering, Design, Analysis and Manufacturing 1(02), pp. 103-108. 1987.

[29] E. L. Rissland, J. J. Daniels, Z. B. Rubinstein and D. B. Skalak. Casebased diagnostic analysis in a blackboard architecture. Presented at PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE. 1993.

[30] E. Shahbazian, J. Duquet and P. Valin. A blackboard architecture for incremental implementation of data fusion applications. Presented at FUSION. 1998.

[31] S. H. Rubin, M. H. Smith and L. Trajkovic. A blackboard architecture for countering terrorism. Presented at Systems, Man and Cybernetics, 2003. IEEE International Conference On. 2003.

[32] J. P. Rice. Poligon: A systems for parallel problem solving. Knowledge Systems Laboratory, Stanford University. Stanford, CA. 1986.

[33] M. Hewett and R. Hewett. A language and architecture for efficient blackboard systems. Presented at Artificial Intelligence for Applications, 1993. Proceedings., Ninth Conference On. 1993.

[34] J. Le Mentec and S. Brunessaux. Improving reactivity in a blackboard architecture with parallelism and interruptions. Presented at Proceedings of the 10th European Conference on Artificial Intelligence. 1992.

[35] E. De Poorter, I. Moerman and P. Demeester. An information driven sensornet architecture. Presented at Sensor Technologies and Applications, 2009. SENSORCOMM'09. Third International Conference On. 2009.

[36] L. E. Parker, B. Kannan, X. Fu and Y. Tang. Heterogeneous mobile sensor net deployment using robot herding and line-of-sight formations. Presented at Intelligent Robots and Systems, 2003 (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference On. 2003.

[37] W. Hu, T. Le Dinh, P. Corke and S. Jha. Outdoor sensornet design and deployment: Experiences from a sugar farm. Pervasive Computing, IEEE 11(2), pp. 82-91. 2012.

[38] W. Fink, J. M. Dohm, M. A. Tarbell, T. M. Hare and V. R. Baker. Nextgeneration robotic planetary reconnaissance missions: A paradigm shift. Planet. Space Sci. 53(14), pp. 1419-1426. 2005.

[39] W. Fink, M. A. Tarbell, R. Furfaro, L. Powers, J. S. Kargel, V. R. Baker and J. Lunine. Robotic test bed for autonomous surface exploration of titan, mars, and other planetary bodies. Presented at Aerospace Conference, 2011 IEEE. 2011. DOI: 10.1109/AERO.2011.5747267.

[40] W. Fink, J. M. Dohm, M. A. Tarbell, T. M. Hare, V. R. Baker, D. SchulzeMakuch, R. Furfaro, A. G. Fairén, T. Ferre and H. Miyamoto. Tierscalable reconnaissance missions for the autonomous exploration of planetary bodies. Presented at Aerospace Conference, 2007 IEEE. 2007.

[41] J. Straub and R. Fevig, "Multi-tier planetary exploration: A new autonomous control paradigm," in Proceedings of the AIAA Space 2012 Conference, Pasadena, CA, USA, 2012.

[42] J. Straub, "Multi-tier exploration concept demonstration mission," in Proceedings of the 2012 Global Space Exploration Conference, Washington, DC, USA, 2012.

[43] J. Straub. A data collection decision-making framework for a multi-tier collaboration of heterogeneous orbital, aerial and ground craft. Presented at Proceedings of the SPIE Defense, Security Sensing Conference. 2013.

[44] P. Corke, R. Peterson and D. Rus. "Networked robots: Flying robot navigation using a sensor net," in Robotics Research 2005.

[45] C. A. Boano, T. Voigt, N. Tsiftes, L. Mottola, K. Römer and M. A. Zúniga. "Making sensornet MAC protocols robust against interference," in Wireless Sensor Networks 2010.

[46] M. L. Fairbairn, I. Bate and J. A. Stankovic. Improving the dependability of sensornets.

[47] J. Tate, I. Bate and S. Poulding. Tuning protocols to improve the energy efficiency of sensornets. Presented at Proceedings of the Fourth UK Embedded Forum. 2008.

[48] A. Gupta and R. A. Uthra. Cluster based approximate data collection in wireless sensor network.

[49] J. Straub. Model based data transmission: Analysis of link budget requirement reduction. Communications and Network 4(4), pp. 278-287. 2012.

[50] J. Straub, "Integrating model-based transmission reduction into a multitier architecture," in Proceedings of the 2013 IEEE Aerospace Conference, Big Sky, MT, USA, 2013.

[51] J. Straub. Fusion of data from multiple sensors with model-based data analysis. Presented at SPIE Defense, Security, and Sensing. 2013.

[52] M. Colby and K. Tumer. Multiagent reinforcement learning in a distributed sensor network with indirect feedback. Presented at Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems. 2013.

[53] C. Zhang and V. Lesser. Coordinating multi-agent reinforcement learning with limited communication. Presented at Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems. 2013.

[54] A. Hamzi, M. Koudil, J. Jamont and M. Occello. Multi-agent architecture for the design of WSN applications. 2013.

[55] B. N. Chun, P. Buonadonna, A. AuYoung, C. Ng, D. C. Parkes, J. Shneidman, A. C. Snoeren and A. Vahdat. Mirage: A microeconomic resource allocation system for sensornet testbeds. 2005.

[56] M. Haghighi. "Market-based resource allocation for energy-efficient execution of multiple concurrent applications in wireless sensor networks," in Mobile, Ubiquitous, and Intelligent Computing 2014.