International Journal of Automation and Computing 12(5), October 2015, 503-510
DOI: 10.1007/s01163-015-0896-4

Comparing the Effect of Pruning on a Best Path and a Naïve-approach Blackboard Solver

Jeremy Straub

Department of Computer Science, University of North Dakota, Grand Forks, North Dakota 58202, USA

Abstract: A naïve solver is one approach that can be used to identify prospective solutions based on data on (or projected to be on) a Blackboard Architecture′s blackboard. The naïve solver approach doesn′t implement heuristics or other techniques to determine what solution paths to attempt first. Instead, it runs the blackboard forward (simulating what would occur if data were gradually added to the blackboard at a faster-than-real time rate). The approach doesn′t guarantee that an optimal solution will be found and will need to be run repetitively to create multiple solutions for comparison. This paper assesses the effect of pre-pruning the blackboard′s facts and rules to remove those that are not relevant (e.g., facts that cannot be asserted, rules that cannot be triggered) or which produce irrelevant facts and pruning actions that produce irrelevant facts (and/or trigger other similarly useless actions). It describes the Blackboard implementation and its utility, explains the pruning process used and presents quantitative and qualitative assessment of the utility of pruning to a naïve solver′s operations. This value is extrapolated to facilitate consideration of a more robust pruning process which also removes low-value facts, actions and rules in addition to those being removed due to their uselessness.

Keywords: Blackboard Architecture, blackboard maintenance, control of heterogeneous craft, multi-craft control, multi-tier control.

1 Introduction

Numerous uses of the Blackboard Architecture[1] have been proposed for a variety of applications. The blackboard approach extends the concept of an expert system to actuate, instead of simply providing recommendations to a human operator. While a blackboard system can be utilized in forward-only mode, where new data is applied and facts or actions are triggered from it, a more proactive approach, for certain applications, is to steer the data that is collected by the blackboard based on an identified best-possible path[2, 3]. This path may not be able to be perfectly followed: For example, if data that is collected is different from what is predicted or if physical impediments prevent collection, an alternate path may be needed. However, this approach can position the system to make the best choices based on currently known information (and update the path and thus the choices made as additional information is added to the blackboard).

In order to identify this best path (based on the present state), a solver must be run on the blackboard. Several different approaches to this can be taken. One approach which closely mirrors the way the blackboard system operates in forward-only mode, is a naïve solver. This solver asserts facts (triggering rules and actions) and continuously checks to see if the desired outcome has been produced. The naïve solver can be run multiple times to identify multiple solutions for comparison. This approach may be particularly well suited to a distributed blackboard system (where there is no single repository of facts, actions and rules) as it will not require every single fact, action and rule to have to be transferred to be assessed. However, the data required may still be significant (though comparatively less).

One way of potentially reducing the amount of time that is required for any solver to run is to prune off unnecessary data (to prevent it from having to be processed). This paper considers the impact of pruning on the performance of the naïve solver. It compares the time saved to the amount of time consumed by pruning, evaluating the value provided quantitatively. A qualitative evaluation of the approach′s utility in a variety of conditions is also presented.

Expediency of blackboard operations is required for many prospective applications. For example, it is critical if a Blackboard architecture is used in the development of new solutions for path planning for unmanned aerial[4], ground[5, 6] and underwater vehicles[7]. Autonomous control of game agents[8] would also drive a need for timely decision making. The evaluation conducted herein, thus, supports assessment of the suitability of pruning best path and naïve solvers for these applications.

2 Background

The work described herein draws on a significant history of prior work related to the Blackboard Architecture, its implementation and its uses, originating from Hayes-Roth′s initial work[1] on refining the Hearsay II system[9]. First,

Regular Paper
Manuscript received September 21, 2013; accepted December 4, 2014
This work was supported by a Grant-In-Aid of Research from Sigma Xi, the Scientific Research Society, North Dakota EPSCoR (NSF # EPS-814442) and a Summer Doctoral Fellowship from the University of North Dakota School of Graduate Studies. Facilities and equipment utilized in this work have been provided by the University of North Dakota Department of Computer Science, and North Dakota EPSCoR (NSF # EPS-814442).
Recommended by Associate Editor Mohammed Chadi
© Institute of Automation, Chinese Academy of Science and Springer-Verlag Berlin Heidelberg 2015

an overview of the Blackboard Architecture will be provided, including a discussion of its previous uses. Then, the use of pruning in systems implementing a Blackboard Architecture will be discussed.

2.1 Blackboard Architecture

The Blackboard Architecture functions are similar to an expert system (see [10, 11]) in that it is comprised of facts (logical statements that can be asserted or not, and in some implementations, asserted with a value assigned) and mechanisms for asserting the truth or falsity of those facts (i.e., actions and rules). Unlike an expert system, a blackboard system also commands actions to actuate outcomes in its environment. Under Hayes-Roth′s initial design[1], based off the Hearsay II system[9], two blackboards were utilized: one for domain problems (specific to the topic of the system) and the other for control problems (how to effect goals). However, subsequent implementations have used a variety of configurations. Velthuijsen et al.[12] proposed a parallel implementation while Ettabaa et al.[13−15] proposed various forms of a distributed Blackboard Architecture. Rosenking and Roth[16] demonstrated how systems can cooperate using the blackboard concept, while Dong et al.[17] showed how an event-based blackboard system can be developed.

A variety of changes and enhancements have been made to the base blackboard concept. Corkill et al.[18, 19] worked to create a generic Blackboard-style architecture and its implementation. Alexander-Craig[20] claimed to have reinterpreted the Blackboard Architecture by adding new task capabilities, message handling and filtering capabilities, among other things. Jiang et al.[21] created an adaptive implementation for use by distributed agents, while Laasri et al.[22] demonstrated reasoning ability in the time and hypothetical domains. Rice et al.[23, 24] created a language for Blackboard problem solving implementation, while Mentec et al.[25−27] demonstrated its utility for real-world control problems.

The utility of Blackboard systems have been demonstrated across a wide variety of domains. Huang et al.[28] demonstrated its utility in an intelligent tutoring system. Campos et al.[29−35] demonstrated the use of Blackboard for robotic control. Adhau et al.[36] demonstrated it effectiveness for project scheduling.

2.2 Blackboard pruning

Among the earliest mentions of the concept of pruning in an expert systems and blackboard context is work by Giuliano and Jones[37] who published a technical report discussing the value of utilizing humans to prune computer-generated association lists in 1966. By 1977, the concept of pruning had been considered for use in conjunction with the Blackboard Architecture by Goodman and Reddy[38]. However, their paper provided only a suggestion of the value of its use. In 1987, Craig[39] utilized pruning to remove aircraft that were not deemed to be impactful from a Blackboard-based airspace monitoring system. This pruning, however, was at the object (and not rule/fact/action) level. It has also been suggested in the context of restricting design choices[40], pruning decision trees[41] and removing aspects of the solution space deemed inferior choices by an evaluation process[42−46]. Ward and Novick[47] discussed the problem of pruning away a desirable answer and Ou et al.[48, 49] discussed prospective solutions to this problem, via a configurable severity parameter and global best-value consensus. The pruning concept has also been used in autonomous control outside of Blackboard-based systems. As one example, Rantanen and Juhola [50] used what they termed "configuration deactivation" for a similar problem of reducing the search space for path planning.

3 Best path and naïve solvers

The best path is taken to be the path that requires the lowest cost (which is a combination of the computational cost of running rules and the costs attributable to actions). In most systems that operate in a real-world environment, the action costs (e.g., the time and fuel used for moving a craft and collecting data) will dwarf the computational costs of rule activation. However, this may not always be the case. Rules requiring particularly robust analysis may take longer than actions which do not have a physical component (e.g., triggering a message to be sent across a network). Also, the level of concurrency possible may impact this comparison as well.

The best path is identified based on predictions related to certain elements. Facts that are asserted can obviously be taken as given. However, the results of actions or rules may be unpredictable (i.e., there would be little point to collecting data which is already absolutely known, thus the results of data collection can be projected based on a prior knowledge and past experiences, but surprises could and should occur). Thus, for the purposes of solving for the best path, the outcomes of actions are predicted. A more complex approach (a subject for prospective future work) would be to evaluate multiple result permutations.

The naïve solver algorithm is depicted in Fig. 1. It begins by selecting an invokable rule (one with all preconditions satisfied) to run (if there is not one, the algorithm ends with no solution found). The rule is then run, which may or may not assert one or more facts and/or trigger one or more actions. Each action that is triggered may trigger additional actions (i.e., recursive chains of actions) and assert one or more facts. Once all facts are asserted and all actions are run, the algorithm checks to see if the designated final condition is reached. If not, the invokable rules are identified and the process restarts with the selection of an invokable rule to run.

4 Experimental design

A Blackboard-style system was implemented incorporating the naïve solver depicted in Fig. 1 and described in the previous section. This implementation also incorporated a pruning engine, which is described in Section 4.1 and

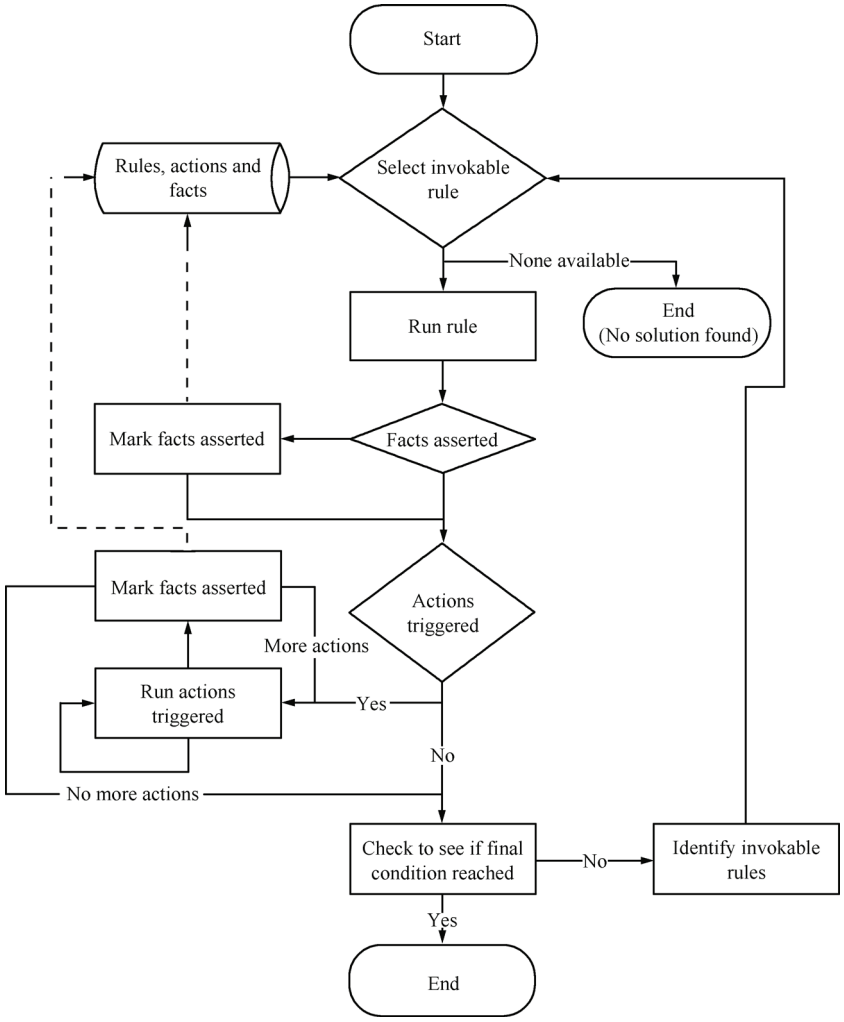

Fig. 1 Naïve solver algorithm

depicted in Fig. 2. Following this, in Section 4.2, the experimental setup and regime are described. Section 5 presents the data that was collected through this process.

4.1 Pruning engine

The pruning engine that was developed operates iteratively. The engine, depicted in Fig. 2, begins by identifying facts that don′t serve as rule conditions and facts that are not currently asserted and which cannot be asserted (e.g., there is no rule or action that asserts them). A placeholder value is then inserted into each rule which requires one of these facts as a precondition and they are removed from the list of facts to be asserted by rules and actions.

Rules that cannot be asserted now (e.g., those with the placeholder values) as well as rules with empty trigger lists are next identified and removed. Finally, actions that are no longer in any triggered list (i.e., which cannot be invoked now) are now identified and deleted. If any change is made during this iteration of the pruning engine, the process restarts (as the changes made may allow other changes to be made). If not, the engine ends.

4.2 Experimental setup and regime

To quantify the time required for the pruning algorithm and to test and compare the performance of the naïve solver using pruned and un-pruned data, 500 trials were run. Each trial began with the creation of a random blackboard configuration. The beginning configuration included 1000 rules, 1000 facts and 1000 actions. For each fact, a random number of prerequisite facts (constrained by a maximum value parameter) was determined and this number of facts were randomly selected for use as prerequisites. For each fact and action, a random number of triggered facts and/or actions (constrained by a maximum value parameter) was determined. Whether a fact or action would be used was then determined randomly for each slot. Finally, the applicable fact or action was randomly selected. A parameter-based number of facts were randomly selected to be initially asserted.

For the non-pruned trials, two steps were then performed. First, an alternate solver was run on the data which is guaranteed to find the best path. This was performed to allow the complexity of trials to be compared quantitatively. Sec-

ond, the naïve solver was run on the blackboard. The results of the trial (presented in Section 5) were recorded and the next trial commenced.

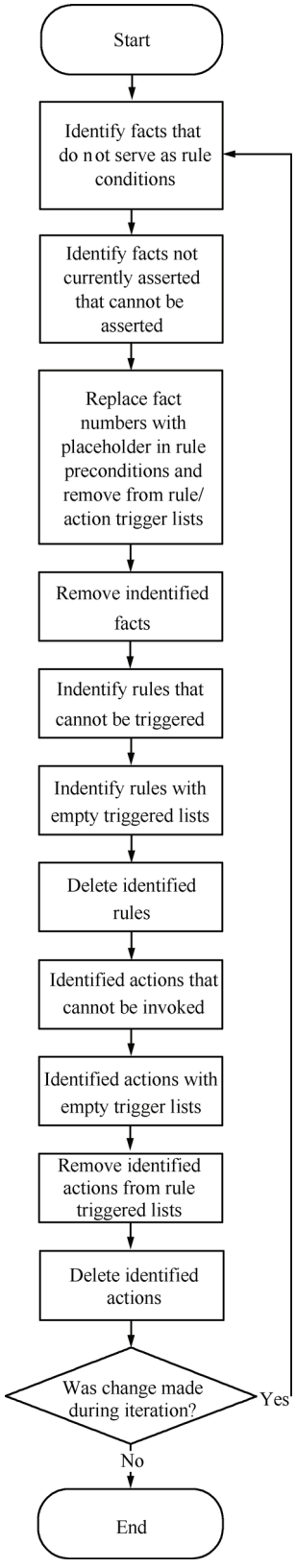

Fig. 2 Pruning engine algorithm

For the pruned trials, the process began by performing the pruning of the blackboard. This process continued iteratively (as described in Section 3) until a run completed with no change being made. The final numbers of facts, rules and actions as well as the amount of time required were recorded for each iteration. Next, the guaranteed-optimal solver was run to allow comparison of the complexity of the solution from run to run. Finally, the naïve solver was run and the results were recorded.

It is important to note that some of the networks produced may not be solvable or that the naïve solver may fail to solve networks in certain cases. The solver automatically gives up after an amount of time that is significantly longer than the time typically required to find a solution.

5 Data collected

This section presents the data collected during the experimentation discussed in Section 4. First, the non-pruned naïve solver results are presented in Table 1. The first four fields present the data (number of iterations, time to populate, time to solve and the path length determined) for the guaranteed-optimal solver. The remaining five fields characterize the performance of the naïve solver. The find count field indicates the numbers of loops of the naïve solver algorithm that were run, the rules run and acts run fields indicate the number of rules and actions invoked, respectively. The time field indicates the total time consumed by the naïve solver and the not found field indicates how many of the 500 trials resulted in no solution being identified.

The data for the pruned naïve solver is divided into two tables for ease of reading. Table 2 provides the data for the pruner algorithm and Table 3 provides the data for the solver. The pruner algorithm′s data (in Table 2) begins with the amount of time that was required for the pruning engine to run. The next three fields indicate the numbers of facts, rules and actions, respectively, which were left when the pruner completed.

In Table 3, the solver results begin with the data related to the guaranteed-optimal solver (which is located in the first four fields). The remaining five fields present the data for the naïve solver. Note that the fields in Table 3 correspond to the field in Table 1 with the same name. Thus, the description of each field will not be repeated.

The point of presenting both the guaranteed solver and naïve approaches is multi-faceted. First, it demonstrates the impact of pruning on both. The guaranteed solver′s time commitment for a non-preprocessed network is actually a combination of the preparation time (i.e., the second column of Tables 1 and 3) and the solve time (third column). This is still less than the naïve solver–across both conditions. However, it is notable that the pruning improves the naïve solver′s performance significantly. This is discussed in greater detail in the subsequent section.

The naïve approach is important, in its own right, in several instances. First, the naïve approach is the typical method used by forward-only Blackboard systems which

Table 1 Non-pruned guaranteed optimal and naïve solver results (mean values from 500 runs)

Guaranteed optimal solver					Naïve solver			
Iter	Time	Solve time	Path length	Find count	Rules run	Acts run	Time	Not solved
7.9	1 197.5	23.4	8.9	33.8	28 793.6	38 039.5	5 680.3	14

Table 2 Pruned naïve solver results, pruner time and results (mean values from 500 runs)

Time	Facts	Rules	Actions
507 906.2	685.6	938.9	667.7

Table 3 Pruned guaranteed optimal and naïve solver results (mean values from 500 runs)

Guaranteed optimal solver					Naïve solver			
Iter	Time	Solve time	Path length	Find count	Rules run	Acts run	Time	Not solved
9.6	1 317.4	20.8	11.6	14.0	12 877.8	17 747.8	2 366.0	6

look for other rules to assert once a new fact is asserted. Second, even in a solving blackboard system (such as the one discussed), the naïve approach serves a role in dealing with dynamic data. Thus, the impact of the pruning on it may be critical for systems that need to perform well during periods where an assumption is violated and an update of the Blackboard network preparations for the guaranteed solver has not yet been performed. Third, there are some network configurations where the naïve solver may outperform the guaranteed one. Characterization of areas of superior naïve solver performance remains a subject for future research.

6 Analysis of data

The data presented in the previous section demonstrates the clear value of the pruning process to the naïve solver. While the performance of the guaranteed-optimal approach does not change significantly (the number of iterations and path length increase slightly, as the population time and the solve time decreases by approximately 11%), the impact on the naïve solver is more pronounced. The naïve solver now only requires 41.3% of the number of iterations that it did previously to generate a solution and it runs only 44.7% of the rules and 46.7% of the actions of the non-pruned approach. The number of instances where a solution could not be identified drops from 2.8% to 1.2%. Perhaps most importantly, the amount of time required decreases to only 41.7% of the non-pruned approach.

The pruner, however, is computationally intensive to run, requiring an average of 507 906.2 ticks. This is, of course, much more than the average savings per solution generated (of 3 315.4 ticks). Thus, to justify the cost of the pruning, at least an average of 153.2 uses of the solver must be run for each pruning. The initial pruning, under the random model presented is (of course) the most expensive and this may be true for many applications. Practically, this means that the benefit of this first expensive pruning may be enjoyed across multiple iterations or re-pruning and solving (with the re-pruning runs taking significantly less time due to having to do less work).

To demonstrate the lower level of cost that may be enjoyed by subsequent prunings, the amount of time required for the first three iterations of the pruner was collected across five trials. In each of these trials, the third iteration did not produce any additional results (though this would not always be the case). This is presented in Table 4. From this, it is clear that low-work prunings are less expensive (requiring approximately one-half of the time of the initial pruning).

Table 4 Comparative cost of pruning iterations

	Iteration 1	Iteration 2	Iteration 3
Ticks	332 487	170 356.8	170 268.4
Percent	49.4%	25.3%	25.3%

7 Qualitative analysis

The actual benefit produced by pruning is, of course, application dependent. So is the cost of the pruning process. These two are, logically, related as more benefit is received via the removal of more items from the blackboard (which also costs more time). The initial configuration of the node network may also reduce (or increase) the number of pruning iterations required. Pruning may also be stopped before the point that was used for this work (a point at which no further changes occur) as part of a cost/benefit assessment of the projected value of continued pruning.

In addition to the comparison of the time savings performed in Section 6, the value of moving work must also be considered. Pruning is an activity that can be conducted on an as-resources-are-available basis, while the benefit can be enjoyed (potentially) during times where performance is critical, such as decision making for a cyberphysical system. The comparative value of the two types of processing time consumed should also be taken into account as part of the analysis process. This relative value is (of course) application-specific and must be considered in the context of a prospective use of the Blackboard architecture.

The impact of change on the blackboard bears consideration in this context. The projection capability of the solving

blackboard approach is enabled by a network of potential facts and their relevance (indicated by rules/actions). For example, if the blackboard held data about a land survey grid, facts that could be asserted might include the presence of various types of objects of interest. Some of these facts may have rules which they satisfy. The projection routine either assumes that all possible facts are true (the simple approach) or projects (from prior knowledge, etc.) the likelihood of a fact being true. Thus, the importance of collecting data to allow the assertion or refutation of a fact can be determined based on its impact on the rest of the network. If a projection turns out to be false, immediate decision making is required. A pre-pruned blackboard can aid in the speed of this decision making.

8 Conclusions and future work

This paper has discussed a naïve solver algorithm for the Blackboard architecture and considered the effect of pruning the blackboard, using a presented pruning algorithm, on the performance of the naïve solver. The pruning approach utilized removed only completely unneeded items. However, the performance differential would be similar for an algorithm that removed a similar percentage of unlikely-to-be-needed (or other heuristically determined set of rules, etc.). The value of the pruning was demonstrated, showing that the pruned blackboard could be solved in less than half of the time of the non-pruned blackboard: it required less than half of the solver iterations, rule executions and action executions. The value of pruned solving was compared to the cost of pruning, demonstrating that approximately 153 uses of the pruned network would be required to cost-justify the pruning solely on this metric. The notion of reducing re-pruning cost was discussed (allowing this initial cost to be spread over extended operations with a significantly lower cost level being incurred for subsequent re-prunings). Also, the value of shifting work from periods that are performance critical to periods where the processing capabilities are underutilized was discussed.

Future work will involve continued refinement of the quantification of the value of pruning. This will include assessment of whether an incomplete pruning may maximize the cost-benefit equation. The value of pruning across multiple different network configurations is also a topic for future work.

References

[1] B. Hayes-Roth. A blackboard architecture for control. *Artificial Intelligence*, vol. 26, no. 3, pp. 251–321, 1985.

[2] J. Straub. A data collection decision-making framework for a multi-tier collaboration of heterogeneous orbital, aerial, and ground craft. In *Proceedings of SPIE 8742, Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR IV*, SPIE, Bellingham, USA, 2013.

[3] J. Straub, H. Reza. The use of the blackboard architecture for a decision making system for the control of craft with various actuator and movement capabilities. In *Proceedings of the 11th International Conference on Information Technology: New Generations*, IEEE, Las Vegas, USA, pp. 514–519, 2014.

[4] J. Clarke, W. H. Chen. Trajectory generation for autonomous soaring UAS. *International Journal of Automation and Computing*, vol. 9, no. 3, pp. 248–256, 2012.

[5] C. J. Liu, W. H. Chen, J. Andrews. Experimental tests of autonomous ground vehicles with preview. *International Journal of Automation and Computing*, vol. 7, no. 3, pp. 342–348, 2010.

[6] T. K. Wang, Q. Dang, P. Y. Pan. Path planning approach in unknown environment. *International Journal of Automation and Computing*, vol. 7, no. 3, pp. 301–316, 2010.

[7] B. K. Sahu, B. Subudhi. Adaptive tracking control of an autonomous underwater vehicle. *International Journal of Automation and Computing*, vol. 11, no. 3, pp. 299–307, 2014.

[8] U. K. Patel, P. Patel, H. Hexmoor, N. Carver. Improving behavior of computer game bots using fictitious play. *International Journal of Automation and Computing*, vol. 9, no. 2, pp. 122–134, 2012.

[9] L. D. Erman, F. Hayes-Roth, V. R. Lesser, D. R. Reddy. The hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty. *ACM Computing Surveys*, vol. 12, no. 2, pp. 213–253, 1980.

[10] D. A. Waterman. *A Guide to Expert Systems*, New York, NY, USA: Addison-Wesley, 1986.

[11] B. Buchanan, D. Barstow, R. Bechtal, J. Bennett, W. Clancey, C. Kulikowski, T. Mitchell, D. Waterman. Constructing an expert system. *Building Expert Systems*, F. Hayes-Roth, D. Waterman, D. Lenat, Eds., New York, USA: Addison-Wesley, vol. 50, pp. 127–167, 1983.

[12] H. Velthuijsen, B. Lippolt, J. Vonk. A parallel blackboard architecture. In *Proceedings of the 3rd International Expert Systems Conference*, Medford, USA, pp. 487–493, 1987.

[13] K. S. Ettabaa, I. R. Farah, B. Solaiman, M. B. Ahmed. Distributed blackboard architecture for multi-spectral image interpretation based on multi-agent system. In *Proceedings of Information and Communication Technologies*, IEEE, Damascus, Syria, pp. 3070–3075, 2006.

[14] J. Palma, R. Marín, M. Balsa, S. Barro, P. Félix. A control model for distributed blackboard architecture based on task structures. In *Proceedings of the International Symposium on Engineering of Intelligent Systems*, pp. 476–483, 2001.

[15] D. L. Larner. A distributed, operating system based, blackboard architecture for real-time control. In *Proceedings of the 3rd International Conference on Industrial and Engineering Applications of Artificial Intelligence and expert Systems*, ACM, New York, USA, vol. 1, pp. 99–108, 1990.

[16] J. P. Rosenking, S. P. Roth. REACT: Cooperating expert systems via a blackboard architecture. In *Proceedings of SPIE 0937, Applications of Artificial Intelligence VI*, SPIE, Bellingham, USA, pp. 143–150, 1988.

[17] J. Dong, S. Chen, J. J. Jeng. Event-based blackboard architecture for multi-agent systems. In *Proceedings of IEEE International Conference on Information Technology: Coding and Computing*, IEEE, Las Vegas, USA, vol. 2, pp. 379–384, 2005.

[18] D. D. Corkill, K. Q. Gallagher, K. E. Murray. GBB: A generic blackboard development system. In *Proceedings of the National Conference on Artificial Intelligence*, Palo Alto, USA, pp. 1008–1014, 1986.

[19] S. D. Tynor, S. P. Roth, J. F. Gilmore. Implementation of a generic blackboard architecture. In *Proceedings of SPIE 0786, Applications of Artificial Intelligence V*, SPIE, Bellingham, USA, pp. 116–124, 1987.

[20] I. D. Alexander-Craig. A New Interpretation of the Blackboard Architecture, Technical Report, University of Warwick, [Online], Available: http://eprints.dcs.warwick.ac.uk/1368/1/cs-rr-254.pdf, 1993.

[21] Y. C. Jiang, Z. Y. Xia, Y. P. Zhong, S. Y. Zhang. An adaptive adjusting mechanism for agent distributed blackboard architecture. *Microprocessors and Microsystems*, vol. 29, no. 1, pp. 9–20, 2005.

[22] H. Laasri, B. Maitre, T. Mondot, F. Charpillet, J. P. Haton. ATOME: A Blackboard Architecture with Temporal and Hypothetical Reasoning, Research Report RR-0855, Inria, Le Chesnay Cedex, France, [Online], Available: http://hal.archives-ouvertes.fr/docs/00/07/71/80/PDF/RR-0855. pdf, 1988.

[23] J. Rice. Poligon: A System for Parallel Problem Solving, Technical Report 86–19, Knowledge Systems Laboratory, Stanford University, Stanford, USA, [Online], Available: http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA170399#page=166, 1986.

[24] M. Hewett, R. Hewett. A language and architecture for efficient blackboard systems. In *Proceedings of the 9th Conference on Artificial Intelligence for Applications*, IEEE, Orlando, USA, pp. 34–40, 1993.

[25] J. Le Mentec, S. Brunessaux. Improving reactivity in a blackboard architecture with parallelism and interruptions. In *Proceedings of the 10th European Conference on Artificial Intelligence*, John Wiley & Sons, Inc., Hoboken, USA, pp. 255–256, 1992.

[26] G. K. H. Pang. A blackboard control architecture for real-time control. In *Proceedings of the American Control Conference*, IEEE, Atlanta, USA, pp. 221–226, 1988.

[27] F. Ingrand, V. Coutance. Procedural reasoning versus blackboard architecture for real-time reasoning. In *Proceedings of the 13th International Conference on Artificial Intelligence*, Avignon, France, pp. 449–459.

[28] M. J. Huang, H. K. Chiang, P. F. Wu, Y. J. Hsieh. A multi-strategy machine learning student modeling for intelligent tutoring systems: Based on blackboard approach. *Library Hi Tech*, vol. 31, no. 2, pp. 274–293, 2013.

[29] A. de Campos, M. J. Monteiro de Macedo. A blackboard architecture for perception planning in autonomous vehicles. In *Proceedings of the International Conference on Power Electronics and Motion Control*, IEEE, San Diego, China, pp. 826–831, 1992.

[30] R. A. F. Romero, E. Prestes, M. A. P. Idiart, G. Faria. Locally oriented potential field for controlling multi-robots. *Communications in Nonlinear Science and Numerical Simulation*, vol. 17, no. 12, pp. 4664–4671, 2012.

[31] S. Rockel, B. Neumann, J. Zhang, K. Dubba, A. Cohn, Š. Konečný, M. Mansouri, F. Pecora, A. Saffiotti, M. Günther, S. Stock, J. Hertzberg, A. M. Tomé, A. Pinho, L. Seabra Lopes, S. Von Riegen, L. Hotz. An ontology-based multi-level robot architecture for learning from experiences. In *Proceedings of the AAAI Spring Symposium on Designing Intelligent Robots: Reintegrating AI II*, Palo Alto, USA, 2013.

[32] N. Michael, E. Stump, K. Mohta. Persistent surveillance with a team of MAVs. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, San Francisco, USA, pp. 2708–2714, 2011.

[33] G. Brzykcy, J. Martinek, A. Meissner, P. Skrzypczynski. Multi-agent blackboard architecture for a mobile robot. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, Maui, USA, vol. 44, pp. 2369–2374, 2001.

[34] S. Carroll, J. E. Boyd, J. Denzinger. Data-centered control of cooperating UAVs: Flying airplanes with a multimedia database, [Online], Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.99.2334&rep=rep1&type=pdf, 2008.

[35] D. A. Goldin, A. M. Chesnokov. Features of informational control complex of autonomous spacecraft. In *Proceedings of IFAC Workshop on Aerospace Guidance, Navigation and Flight Control Systems*, IFAC, Samara, Russia, 2011.

[36] S. Adhau, M. L. Mittal, A. Mittal. A multi-agent system for distributed multi-project scheduling: An auction-based negotiation approach. *Engineering Applications of Artificial Intelligence*, vol. 25, no. 8, pp. 1738–1751, 2012.

[37] V. E. Giuliano, P. F. Jones. Study and Test of a Methodology for Laboratory Evaluation of Message Retrieval Systems, Government Report #ad0642829, USA, 1966.

[38] G. Goodman, R. Reddy. Alternative Control Structures for Speech Understanding Systems, [Online], Available: http://repository.cmu.edu/cgi/viewcontent.cgi?article=3374&context=compsci, 1977.

[39] I. Craig. CASSANDRA-II: A Distributed Blackboard System, Technical Report RR90, University of Warwick, USA, 1987.

[40] S. Bhansali, H. P. Nii. KASE: An integrated environment for software design. *Artificial Intelligence in Design'92*, J. S. Gero, F. Sudweeks, Eds., Netherlands: Springer, pp. 371–389, 1992.

[41] B. A. Draper, A. R. Hanson, E. M. Riseman. Learning blackboard-based scheduling algorithms for computer vision. *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 7, no. 2, pp. 309–328, 1993.

[42] B. Hayes-Roth, A. Collinot. A satisficing cycle for real-time reasoning in intelligent agents. *Expert Systems with Applications*, vol. 7, no. 1, pp. 31–42, 1994.

[43] K. S. Decker, V. R. Lesser, R. C. Whitehair. Extending a blackboard architecture for approximate processing. *Real-Time Systems*, vol. 3, no. 1–2, pp. 47–79, 1990.

[44] K. S. Decker, V. R. Lesser, R. C. Whitehair. Extending a Blackboard Architecture for Approximate Processing, COINS Technical Report, Computer and Information Science Department, University of Massachusetts, Amherst, USA, 89–115, 1990.

[45] A. Kemper, G. Moerkotte, K. Peithner, M. Steinbrunn. Optimizing disjunctive queries with expensive predicates. In *Proceedings of ACMS IGMOD International Conference on Management of Data*, vol. 23, no. 2, pp. 336–347, 1994.

[46] C. Alvarado, M. Oltmans, R. Davis, A. Davis. A framework for multi-domain sketch recognition. In *Proceedings of the AAAI Spring Symposium on Sketch Understanding*, Palo Alto, USA, 2002.

[47] K. Ward, D. Novick. On the Need for a Theory of Knowledge Sources for Spoken-Language Understanding. In *Proceedings of AAAI Workshop on Integration of Natural Language and Speech Processing*, AAAI, Seattle, USA, pp 23–30, 1994.

[48] S. Y. Ou, C. S. G. Khoo, D. H. Goh. Design and development of a concept-based multi-document summarization system for research abstracts. *Journal of Information Science*, vol. 34, no. 3, pp. 308–326, 2008.

[49] F. Stahl, M. Bramer, M. Adda. J-PMCRI: A methodology for inducing pre-pruned modular classification rules. *Artificial Intelligence in Theory and Practice III*, M. Bramer, Ed., Berlin, Germany: Springer, pp. 47–56, 2010.

[50] M. Rantanen, M. Juhola. A configuration deactivation algorithm for boosting probabilistic roadmap planning of robots. *International Journal of Automation and Computing*, vol. 9, no. 2, pp. 155–164, 2012.

Jeremy Straub received bachelor′s degrees in business and information technology, an M. Sc. degree in computer systems and software design from Jacksonville State University, USA and an MBA from Mississippi State University, USA. He has published over 30 journal articles and 100 full conference papers. He has also authored over 55 presentations at national or international conferences and 120 at local and regional conferences.

His research interests include development, implementation and assessment of autonomous control technologies to answering questions of policy and law about how technology should be utilized.

E-mail: jeremy.straub@my.und.edu

ORCID iD: 0000-0002-9821-2858