Creating A Uniform Framework for Task and Motion Planning: A Case for Incremental Heuristic Search? [Overview Paper]

Sven Koenig
Department of Computer Science
University of Southern California
Los Angeles, CA 90089-0781
skoenig@usc.edu

Abstract

In this short overview paper, we describe our vision for combining task and motion planning and present a historical perspective to show which parts of it have already become reality. Robots do not have to plan only once but repeatedly. Replanning from scratch is often very time consuming. Incremental heuristic search addresses this issue by reusing information from previous searches to find solutions to series of similar search tasks often much faster than is possible by solving each search task from scratch. Incremental heuristic search has mostly been used for path planning in the past but we argue that it applies to many layers of robot architectures, in particular task and motion planning, which might allow one to design very homogeneous robot architectures with clean interfaces between the layers.

Introduction

Mobile robot architectures typically consist of several layers which suitably partition system functionality, see Figure 1. For example, office delivery robots have to determine in which order to visit offices, plan paths to those offices, follow those paths reliably and avoid static and dynamic obstacles in the process. Task planning determines where the robot should move next, while motion planning moves the robot there. The conventional wisdom in the 1990s was the following: For complex motions, such as parallel parking, one uses a full-scale motion planner. For simple motions, such as obstacle avoidance, one uses a combination of path planning and navigation (that is, path following). Path planning searches a low-dimensional configuration space (for example, given by the coordinates of the robot) with medium time lookaheads to determine the nominal trajectory of the robot, often ignoring obstacles and the physical capabilities of the robot (such as dynamic constraints). Navigation searches a high-dimensional configuration space (for example, given by the coordinates of the robot, its orientation and its velocity) with smaller time lookaheads to follow the nominal trajectory as closely as possible, taking obstacles and the physical capabilities of the robot into account. The conventional wisdom in the 1990s was that task planning is best implemented with symbolic planning methods, path planning is best implemented with search methods, and navigation is best implemented with reactive navigation methods. Xavier (Simmons et al. 1997), for example, was an indoor mobile robot that operated from December 1995 for more than four years. It accepted about 40,000 requests from the world wide web to visit different offices, traveling about 240 kilometers in the process (Simmons et al. 2000). Task planning used Prodigy (Carbonell et al. 1991) to process incoming navigation requests, prioritize them, and identify when different navigation requests could be achieved opportunistically. Path planning used a version of the heuristic search method A* (Hart, Nilsson, and Raphael 1968) to determine efficient routes based on a topological map augmented with rough metric information, taking into account that some paths can be followed more easily than others. Navigation used the Lane-Curvature Method (Ko and Simmons 1998) to find highly traversable lanes in the desired direction and the Curvature-Velocity Method (Simmons 1996) to switch between lanes and avoid dynamic obstacles. Servo control then controlled the motors appropriately. This description is somewhat simplified since both path planning and navigation used partially observable Markov decision problem-based robot navigation (Koenig and Simmons 1998) to deal with actuator and sensor uncertainty and the resulting position uncertainty. In general, the lower layers of a robot architecture typically have to process lots of low-level (raw) data under extreme time pressure to solve simple planning problems with high frequency using an extremely local view, while higher layers typically have to process a smaller amount of aggregated high-level (abstract) data under less time pressure to solve more difficult planning tasks with lower frequency using a more global view.

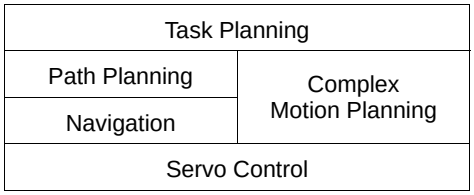

Figure 1: Simple Robot Architecture

We argue in this overview paper that developments in the past decade have made it possible to implement not only path planning but also task planning, motion planning for complex motions and navigation with versions of A*, except possibly for the runtime requirements. Robots need to plan on-line to be responsive to the current naviga-

tion scenario. They do not have to plan only once but repeatedly because their actuation and sensing are imperfect, their knowledge of the domain is imperfect or the domain changes dynamically. For example, task planning has to replan when a new task is introduced. Path planning has to replan when an unmodeled obstacle blocks the current path to the existing destination or task planning changes the destination. Navigation has to replan when the robot needs to circumnavigate an unmodeled obstacle or when path planning changes the path. Task planning solves difficult and thus time-consuming planning tasks. Navigation solves simpler and thus less time-consuming planning tasks but needs to be run with high frequency. It thus needs to be extremely fast to provide safe motion. Incremental heuristic search speeds up repeated planning by reusing information from previous searches to find solutions to series of similar search tasks often much faster than is possible by solving each search task from scratch. Incremental heuristic search has mostly been used for path planning in the past but we argue that it applies to task planning, motion planning for complex motions and navigation as well, which might allow one to design very homogeneous robot architectures with clean interfaces between the layers.

Incremental Heuristic Search

Incremental heuristic search (Koenig et al. 2004) combines incremental and heuristic search. Incremental search reuses information from previous searches to find solutions to series of similar search tasks potentially much faster than is possible by solving each search task from scratch, while heuristic search uses heuristic values that approximate the goal distances to focus the search and solve search problems potentially much faster than uninformed search methods. Incremental and heuristic search have been studied independently since the late 1960s. Incremental heuristic search started with (focussed) D* (Stentz 1995) in the mid-1990s. Three different classes of incremental heuristic search methods are known, all of which are incremental versions of A*:

- The first class restarts A* at the point where its current search deviates from the previous one. Examples are Differential A* (Trovato and Dorst 2002) and FSA* (Sun and Koenig 2007), which are very simple incremental heuristic search methods.
- The second class updates the heuristic values from the previous search during the current search to make them more informed. Examples are Adaptive A* (Koenig and Likhachev 2006), which uses a principle described in the context of Hierarchical A* (Holte et al. 1996), and its generalization Generalized Adaptive A* (Sun, Koenig, and Yeoh 2008).
- The third class updates the g-values from the previous search during the current search to correct them when necessary, which can be interpreted as transforming the A* search tree from the previous search into the A* search tree for the current search. Examples are LPA* (Koenig, Likhachev, and Furcy 2004), D* and D* Lite (Koenig and Likhachev 2002), which is easier to understand than D*.

All these incremental heuristic search methods differ from other replanning methods (such as planning by analogy) in that the resulting path length can be as good as that achieved by planning from scratch (if desired). Thus, the path length does not deteriorate with the number of replanning episodes. Extensions include search with limited resources (such as time and energy constraints) (Mills-Tettey, Stentz, and Dias 2006), search in the presence of position uncertainty (Gonzalez and Stentz 2008), minimax search (Likhachev and Koenig 2003), anytime search (Likhachev et al. 2008) and anyangle search (Ferguson and Stentz 2006; Nash, Koenig, and Likhachev 2009). Incremental heuristic search methods (most notably D*, D* Lite and their extensions) have been used as part of a variety of robotics applications by a number of research groups. They are typically applied to path planning in the context of moving a robot to given goal coordinates in initially unknown terrain. Planning with the freespace assumption assumes that the terrain is clear unless it knows otherwise. It always plans a shortest path to the given goal coordinates and replans whenever it detects that its current path is no longer optimal (for example, because it is blocked by an obstacle), resulting in an effective and efficient robot navigation method in initially unknown terrain (Koenig, Smirnov, and Tovey 2003). We now explain how incremental heuristic search applies to task and motion planning.

Task Planning

Task planning searches large state spaces with large time lookaheads to determine where the robot should move next. Symbolic planners are often based on heuristic search. Heuristic search-based planners were introduced in (McDermott 1996) and (Bonet, Loerincs, and Geffner 1997) in the mid-1990 and have become very popular since then. HSP 2.0 (Bonet and Geffner 2000), for example, uses weighted A* (Pohl 1970) with inadmissible heuristic values to perform forward searches in the space of world states to find a path from the current state to a given goal state. This is possible despite the large state spaces due to the specific heuristic values used. Since task planners are often based on heuristic search, incremental heuristic search can speed them up in two orthogonal ways:

- Incremental heuristic search can speed up solving single heuristic search-based planning tasks. Heuristic search-based planners spend about eighty percent of their planning time on calculating the heuristic values. For example, HSP 2.0 calculates each heuristic value by solving a relaxed planning problem with a dynamic programming method similar to value iteration. PINCH (Liu, Koenig, and Furcy 2002) orders the value updates and reuses information from the calculation of previous heuristic values to speed up the planning time of HSP 2.0 by up to eighty percent in several domains.
- Incremental heuristic search can also speed up solving series of similar heuristic search-based planning tasks. SHERPA (Koenig, Furcy, and Bauer 2002) uses LPA* to speed up solving series of similar planning tasks with HSP 2.0 with admissible heuristic values by up to eighty percent in several domains.

Motion Planning

Motion planning searches high-dimensional configuration spaces (for example, given by the coordinates of the robot, its orientation and its velocity) with medium time lookaheads to determine a trajectory of the robot, taking obstacles and the physical capabilities of the robot into account. Incremental heuristic search in the 2000s is probably more important for motion planning than path planning. Path planning is often fast because it searches a low-dimensional and thus small configuration space, while the robot is often slow not only because it takes time to figure out how to follow the path as closely as possible but mostly because the path does not take the physical capabilities of the robot into account and thus cannot be traversed fast. On the other hand, motion planning is often slow because it searches a high-dimensional and thus large configuration space, while the robot is often fast because the motions take the physical capabilities of the robot into account. and thus can be executed directly. Thus, the time pressure is much more severe for motion planning than path planning.

Complex Motions

For complex motions, such as parallel parking, one might use a full-scale motion planner, even though this is slow. Current motion planners are often based on roadmaps (Kavraki et al. 1996; LaValle and Kuffner 2001). To demonstrate how incremental heuristic search applies to motion planning, we have applied it to motion planning with cell decompositions. Uniform discretizations of the configuration space can prevent motion planning from finding a plan if they are too coarse-grained and result in large state spaces that cannot be searched efficiently if they are too fine-grained. Thus, researchers discretize the configuration space dynamically using nonuniform discretizations. An example is the parti-game method (Moore and Atkeson 1995), a reinforcement-learning method that starts with a coarse discretization and refines it during execution by splitting cells only when and where it is needed (for example, around obstacles). The parti-game method executes a minimax search after each such split. Minimax LPA* (Likhachev and Koenig 2003) is probably the first incremental heuristic search method for search in non-deterministic state spaces. It is an extension of LPA* that speeds up repeated minimax searches and results in an efficient implementation of the parti-game method, which can be an order of magnitude faster than the standard implementation with uninformed search from scratch.

Simple Motions

For simple motions, such as obstacle avoidance, one often uses a combination of path planning and navigation to speed up motion planning. Path planning searches a low-dimensional configuration space (for example, given by the coordinates of the robot) with medium time lookaheads to determine the nominal trajectory of the robot, often ignoring obstacles and the physical capabilities of the robot. Thus, path planning runs sufficiently fast if implemented with search methods. Navigation searches a high-dimensional configuration space (for example, given by the coordinates of the robot, its orientation and its velocity) with smaller time lookaheads to follow the nominal trajectory as closely as possible, taking obstacles and the physical capabilities of the robot into account. Conventional wisdom in the 1990 was that navigation does not run sufficiently fast if implemented with search methods and thus is best implemented with greedy reactive navigation methods, including potential field methods (Koditschek 1989), often implemented as motor schemata grouped into behaviors (Arkin 1989). Reactive navigation is fast but the robot can get stuck in local minima of the potential function, such as in box canyons or in front of small openings. These local minima can be avoided either by sequencing behaviors or by adjusting parameters of the only behavior, either before execution (programming) or during execution (learning). Programming is time intensive and the programmer needs to know the terrain characteristics approximately, otherwise navigation performance can be poor. Learning needs time, both to detect when the parameters should be changed and to determine how to change them (for example, by experimentation), which can degrade the navigation performance. Search can result in a better navigation performance than reactive navigation in the presence of local minima, as shown in Figure 2 (Ranganathan and Koenig 2003) for reactive navigation with learning momentum (Lee and Arkin 2001) (left) and search (right) in a MissionLab simulation (Georgia Tech Mobile Robot Laboratory 2002) in initially unknown terrain with a box canyon. However, search then needs to be run with high frequency. It thus needs to run extremely fast to provide safe motion. Incremental search can be used to speed up path planning but it was unclear whether it can speed up path planning sufficiently to replace reactive navigation.

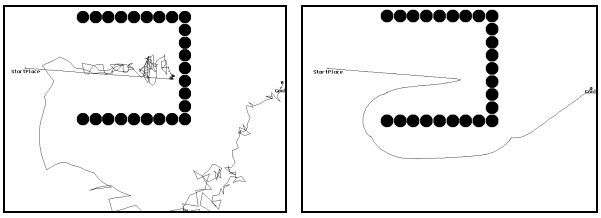

Figure 2: Advantage of Search

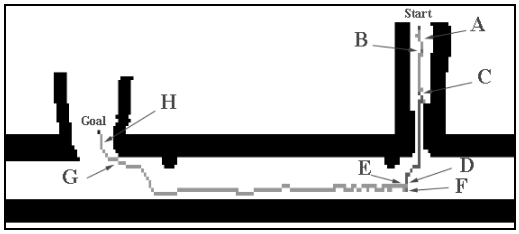

Figure 3: Planning on Demand

Early Proof of Concept: Planning on Demand

We built the Planning on Demand (POD) robot architecture (Ranganathan and Koenig 2003) in 2003, a very simple prototype system to demonstrate that incremental heuris-

tic search can be fast enough to build robot architectures that give path planning progressively greater control of a robot if reactive navigation continues to fail, until path planning controls the robot directly. The POD robot architecture contains a reactive layer (that implements reactive navigation), degenerated sequencing layer (that determines the navigation mode) and powerful deliberative layer (that implements the path planner). The reactive and sequencing layers run continuously but the deliberative layer runs only in certain navigation modes. The reactive layer uses a reactive controller with only one motor schema (whose parameters are not modified during execution) to move the robot to given goal coordinates. This motor schema implements a behavior that consists of two primitive behaviors, namely moving to the destination and avoiding obstacles. The deliberative layer obtains sensor readings from the onboard sensors, updates a short-term map (occupancy grid) and then uses D* Lite for planning with the freespace assumption. The sequencing layer monitors the progress of the robot and determines the navigation mode in a principled way. The amount of path planning and how closely the path planner controls the robot depend on the difficulty that reactive navigation has with the terrain. The sequencing layer uses reactive navigation as much as possible because of its speed (Mode 1). However, reactive navigation can get stuck in box canyons or in front of small openings. If the robot does not make progress toward the destination, then the sequencing layer activates the path planner, which sets a waypoint for reactive navigation to achieve (Mode 2), as had been done before (Wettergreen et al. 2001; Urmson, Simmons, and Nesnas 2003). Reactive navigation can still get stuck if the reactive layer is unable to reach the waypoint. If the robot does not make progress toward the next waypoint, the sequencing layer bypasses reactive navigation completely and lets the path planner control the robot directly for a short time (Mode 3), which is rather unusual in robotics and a first step towards integrating planning more tightly into the control-loop of mobile robots. Figure 3 (Ranganathan and Koenig 2003) shows an example of corridor navigation in initially unknown terrain, including passing through doors. The robot started in Mode 1, entered Mode 2 at point A, Mode 3 at point C, Mode 2 at point D, Mode 1 at point F, Mode 2 at point G and finally Mode 1 at point H. The other points mark additional positions at which the path planner was invoked in Mode 2 to set a waypoint.

Current State of the Art

The POD robot architecture was a radical departure from the thinking that controlling robots directly should be avoided (Gat 1998) and that plans should provide advice but not commands (Agre and Chapman 1990), which was based on experience with classical planning technology that was too slow for researchers to integrate it successfully into the control loop of robots (Fikes and Nilsson 1971). The POD robot architecture was tested only on a slowly moving indoor robot and the results thus do not transfer to fast moving robots, such as unmanned ground vehicles. However, incremental heuristic search has recently been used for motion planning in fielded systems on unmanned ground vehicles. Boss, the winning entry into the Urban Challenge by Carnegie Mellon University (that the author was not involved with), used a hybrid of the two approaches to motion planning presented earlier. Boss used Anytime D* (Likhachev et al. 2008), an extension of D* Lite, on a multi-resolution lattice to plan complex motions in real time (Ferguson, Howard, and Likhachev 2008b), such as the traversal of parking lots and complex u-turns, taking obstacles and the physical capabilities of the robot into account. Incremental heuristic search ran at a frequency of about 1-5 Hertz. Boss still used an impoverished form of reactive navigation (Ferguson, Howard, and Likhachev 2008a), implemented as a local planner that rolled out several possible short trajectories and picked the one that followed the motion produced by the motion planner as closely as possible. Reactive navigation ran at a frequency of 10 Hertz. As envisioned by us, incremental heuristic search allowed the motion planner to take obstacles and the physical capabilities of the robot into account in real-time, which also simplifies reactive navigation. Both motion planning and reactive navigation consider trajectories in a very similar format, which results in robust navigation and avoids situations where the robot gets stuck due to a mismatch between motion planning and navigation. Thus, the vision that incremental search enables one to use search in most layers of robot architectures has partially become true already.

Future Work

Incremental search benefits from two trends, namely that computers get faster and that better sensors and more sophisticated map building techniques create accurate maps fast. However, some technical advances to incremental heuristic search are needed to make all of the envisioned robot architecture a reality. For task planning, incremental heuristic search needs to deal better with innadmissible heuristic values. A starting point exists (Likhachev and Koenig 2005) but there is a tension between incremental heuristic search (that benefits from keeping as much information in memory as possible to speed up future searches) and weighted A* (that reduces the amount of information in memory). For task and motion planning, incremental heuristic search needs to deal better with actuator and sensor uncertainty and the resulting position uncertainty. Starting points exist (Likhachev and Koenig 2003; Gonzalez and Stentz 2008) but incremental heuristic search cannot yet handle general totally or partially observable Markov decision problems. The next step then is to build the complete robot architecture based on ideas from hierarchical search.

Acknowledgments

This overview of our prior research was supported by, or in part by, NSF under contract/grant number 0413196, ARL/ARO under contract/grant number W911NF-08-1-0468 and ONR in form of a MURI under contract/grant number N00014-09-1-1031. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations, agencies or the U.S. government. The overview is a collage

of our prior publications and re-uses text and figures from them and our web pages idm-lab.org. Our description of Xavier follows (Simmons et al. 1997; 2000), our description of incremental heuristic search follows (Koenig et al. 2004), our description of planning for complex motions and of Mini-iMax LPA* follows (Likhachev and Koenig 2003) and our description of planning for simple motions and of the POD robot architecture follows (Ranganathan and Koenig 2003). We acknowledge the research done by the co-authors on our prior publications, given in the bibliography, and additional information on Boss provided by Maxim Likhachev.

References

Agre, P., and Chapman, D. 1990. What are plans for? *Journal for Robotics and Autonomous Systems* 6:17–34.

Arkin, R. 1989. Motor schema-based mobil robot navigation. *International Journal of Robotic Research* 8(4):92–112.

Bonet, B., and Geffner, H. 2000. Heuristic Search Planner 2.0. *Artificial Intelligence Magazine* 22(3):77–80.

Bonet, B.; Loerincs, G.; and Geffner, H. 1997. A fast and robust action selection mechanism for planning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 714–719.

Carbonell, J.; Etzioni, O.; Gil, Y.; Joseph, R.; Knoblock, C.; Minton, S.; and Veloso, M. 1991. PRODIGY: An integrated architecture for planning and learning. *SIGART Bulletin* 2(4):51–55.

Ferguson, D., and Stentz, A. 2006. Using interpolation to improve path planning: The Field D* Algorithm. *Journal of Field Robotics* 23(2):79–101.

Ferguson, D.; Howard, T.; and Likhachev, M. 2008a. Motion planning in urban environments: Part I. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, 1063–1069.

Ferguson, D.; Howard, T.; and Likhachev, M. 2008b. Motion planning in urban environments: Part II. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, 1070–1076.

Fikes, R., and Nilsson, N. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2:189–208.

Gat, E. 1998. On three-layer architectures. In Kortenkamp, D.; Bonasso, R.; and Murphy, R., eds., *Artificial Intelligence Based Mobile Robotics: Case Studies of Successful Robot Systems*. AAAI Press. 195–210.

Georgia Tech Mobile Robot Laboratory. 2002. MissionLab: User manual for MissionLab, Version 5.0. Technical report, College of Computer Science, Georgia Institute of Technology, Atlanta (Georgia).

Gonzalez, J., and Stentz, A. 2008. Replanning with uncertainty in position: Sensor updates vs. prior map updates. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1806–1813.

Hart, P.; Nilsson, N.; and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics* SSC4 4(2):100–107.

Holte, R.; Mkadmi, T.; Zimmer, R.; and MacDonald, A. 1996. Speeding up problem solving by abstraction: A graph oriented approach. *Artificial Intelligence* 85(1-2):321–361.

Kavraki, L.; Svestka, P.; Latombe, J.-C.; and Overmars, M. 1996. Probabilistic roadmaps for path planning in high-dimensional configuration space. *IEEE Transactions on Robotics and Automation* 12(4):566–580.

Ko, N., and Simmons, R. 1998. The Lane-Curvature Method for local obstacle avoidance. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, 1615–1621.

Koditschek, D. 1989. Robot planning and control via potential functions. In Khatib, O.; Craig, J.; and Lozano-Perez, T., eds., *The Robotics Review*. MIT Press. 349–367.

Koenig, S., and Likhachev, M. 2002. Improved fast replanning for robot navigation in unknown terrain. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 968–975.

Koenig, S., and Likhachev, M. 2006. A new principle for incremental heuristic search: Theoretical results [poster abstract]. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 402–405.

Koenig, S., and Simmons, R. 1998. Xavier: A robot navigation architecture based on Partially Observable Markov Decision Process Models. In Kortenkamp, D.; Bonasso, R.; and Murphy, R., eds., *Artificial Intelligence Based Mobile Robotics: Case Studies of Successful Robot Systems*. AAAI Press. 91–122.

Koenig, S.; Likhachev, M.; Liu, Y.; and Furcy, D. 2004. Incremental heuristic search in artificial intelligence. *Artificial Intelligence Magazine* 25(2):99–112.

Koenig, S.; Furcy, D.; and Bauer, C. 2002. Heuristic search-based replanning. In *Proceedings of the International Conference on Artificial Intelligence Planning and Scheduling (AIPS)*, 294–301.

Koenig, S.; Likhachev, M.; and Furcy, D. 2004. Lifelong Planning A*. *Artificial Intelligence Journal* 155(1–2):93–146.

Koenig, S.; Smirnov, Y.; and Tovey, C. 2003. Performance bounds for planning in unknown terrain. *Artificial Intelligence Journal* 147(1–2):253–279.

LaValle, S., and Kuffner, J. 2001. Rapidly-Exploring Random Trees: Progress and prospects. In Donald, B.; Lynch, K.; and Rus, D., eds., *Algorithmic and Computational Robotics: New Directions*. A K Peters. 293–308.

Lee, J., and Arkin, R. 2001. Learning momentum: Integration and experimentation. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 1975–1980.

Likhachev, M., and Koenig, S. 2003. Speeding up the Parti-Game Algorithm. In *Advances in Neural Information Processing Systems (NIPS)*, 1563–1570.

Likhachev, M., and Koenig, S. 2005. A generalized framework for Lifelong Planning A*. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 99–108.

Likhachev, M.; Ferguson, D.; Gordon, G.; Stentz, A.; and Thrun, S. 2008. Anytime search in dynamic graphs. *Artificial Intelligence* 172(14):1613–1643.

Liu, Y.; Koenig, S.; and Furcy, D. 2002. Speeding up the calculation of heuristics for heuristic search-based planning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 484–491.

McDermott, D. 1996. A heuristic estimator for means-ends analysis in planning. In *Proceedings of the International Conference on Artificial Intelligence Planning and Scheduling (ICAPS)*, 142–149.

Mills-Tettey, G.; Stentz, A.; and Dias, B. 2006. DD* Lite: Efficient incremental search with state dominance. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Moore, A., and Atkeson, C. 1995. The Parti-Game Algorithm for variable resolution reinforcement learning in multidimensional state-spaces. *Machine Learning* 21(3):199–233.

Nash, A.; Koenig, S.; and Likhachev, M. 2009. Incremental Phi*: Incremental any-angle path planning on grids. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1824–1830.

Pohl, I. 1970. Heuristic search viewed as path finding in a graph. *Artificial Intelligence* 1:193–204.

Ranganathan, A., and Koenig, S. 2003. A reactive robot architecture with planning on demand. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 1462–1468.

Simmons, R.; Goodwin, R.; Haigh, K.; Koenig, S.; and O'Sullivan, J. 1997. A layered architecture for office delivery robots. In *Proceedings of the International Conference on Autonomous Agents (AGENTS)*, 245–252.

Simmons, R.; Fernandez, J.; Goodwin, R.; Koenig, S.; and O'Sullivan, J. 2000. Lessons learned from Xavier. *IEEE Robotics and Automation Magazine* 7(2):33–39.

Simmons, R. 1996. The Curvature-Velocity Method for local obstacle avoidance. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 3375–3382.

Stentz, A. 1995. The Focussed D* Algorithm for real-time replanning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1652–1659.

Sun, X., and Koenig, S. 2007. The Fringe-Saving A* Search Algorithm - a feasibility study. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2391–2397.

Sun, X.; Koenig, S.; and Yeoh, W. 2008. Generalized Adaptive A*. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 469–476.

Trovato, K., and Dorst, L. 2002. Differential A*. In *IEEE Transactions on Knowledge and Data Engineering*, 1218–1229.

Urmson, C.; Simmons, R.; and Nesnas, I. 2003. A generic framework for robotic navigation. In *Proceedings of the IEEE Aerospace Conference*.

Wettergreen, D.; Shamah, B.; Tompkins, P.; and Whittaker, W. 2001. Robotic planetary exploration by sun-synchronous navigation. In *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation*.