Lifelong Planning for Mobile Robots

Maxim Likhachev and Sven Koenig

College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280, USA
{mlikhach, skoenig}@cc.gatech.edu

Abstract

Mobile robots often have to replan as their knowledge of the world changes. Lifelong planning is a paradigm that allows them to replan much faster than with complete searches from scratch, yet finds optimal solutions. To demonstrate this paradigm, we apply it to Greedy Mapping, a simple sensor-based planning method that always moves the robot from its current cell to the closest cell that it has not observed yet, until the terrain is mapped. Greedy Mapping has a small mapping time, makes only action recommendations and can thus coexist with other components of a robot architecture that also make action recommendations, and is able to take advantage of prior knowledge of parts of the terrain (if available). We demonstrate how a robot can use our lifelong-planning version of A to repeatedly determine a shortest path from its current cell to the closest cell that it has not observed yet. Our experimental results demonstrate the advantage of lifelong planning for Greedy Mapping over other search methods. Similar results had so far been established only for goal-directed navigation in unknown terrain.*

1 Introduction

Artificial intelligence systems have to adapt their plans continuously to changes in the world or their models of the world. Examples include internet agents and mobile robots. Many systems replan from scratch. However, this can be very inefficient in large domains with frequent changes and thus keeps the systems idle, which is often unacceptable. Fortunately, the changes are usually small. For example, an internet agent might learn that a network node went down or a robot might sense a few previously unknown obstacles. This suggests that a complete recomputation of the best plan is wasteful since some of the previous planning results can be reused. Thus, we are developing what we call lifelong planning techniques, in analogy to lifelong learning [35]. Lifelong planning replans faster than planning from scratch because it reuses information from previous plan-construction processes to avoid the parts of the new plan-construction process that are identical to the previous one. We recently introduced lifelong planning methods that are guaranteed to find optimal plans and thus differ from most other methods that make use of previous experience, including case-based planning, plan adaptation, repair-based planning, and learning search-control knowledge. The difficulty is to identify quickly which parts of the previous plan-construction process can be reused.

In this paper, we use lifelong planning on mobile robots because they often have to replan as their knowledge of the world changes. Researchers from mobile robotics have, so far, exploited lifelong planning only for goal-directed navigation in unknown terrain, where Focussed D* [34] convincingly demonstrates its advantages since it solves this search task with a speedup of one to two orders of magnitude(!) over repeated A* searches, which is important to avoid the robots being idle. However, Focussed D* is very complex and thus hard to understand, analyze, and extend. For example, while Focussed D* has been widely used as a black-box method, it has not been extended by other researchers. We believe that two achievements are required to make lifelong planning techniques more popular in mobile robotics. First, one needs to devise simpler lifelong planning techniques. In previous work, we have therefore developed D* Lite [16]. It is simpler and consequently easier to understand and analyze than Focussed D*, yet is at least as efficient. Second, one needs to demonstrate the advantages of lifelong planning methods, such as D* Lite, for additional navigation tasks. In this paper, we therefore demonstrate how D* Lite can be applied to mapping, which is the first demonstration of how lifelong planning methods can be used in the context of mapping. By demonstrating the versatility and computational benefits of lifelong planning for mobile robots, we hope that this underexploited technique will be used more often in mobile robotics.

2 Our Lifelong-Planning Methods

We introduced the term lifelong planning here for the first time. Although lifelong planning methods are not widely known in artificial intelligence and control, some researchers have developed lifelong planning versions of uninformed search methods in the algorithms literature under the name "incremental search." An overview can be found in [14]. We, on the other hand, have developed lifelong-planning versions of A*, thus combining ideas from the algorithms literature and the artificial intelligence literature.

- **Lifelong Planning A*:** For the task of repeatedly finding a shortest path from a given start vertex to a given goal vertex while the edge cost of a graph change, we have developed Lifelong Planning A* [15]. Lifelong Planning A* combines ideas from DynamicSWSF-FP [28] and A* [26]. The first search of Lifelong Planning A* is the same as that of A* but all subsequent searches are much faster. Lifelong Planning A* produces at least the search tree that A* builds. However, it achieves a substantial speedup over A* because they reuse those parts of the previous search tree that are identical to the new search tree, and uses a clever way of identifying these parts. The simplicity of Lifelong Planning A* allows us to prove various properties about it and show a strong similarity to A*.
- **D* Lite:** For the task of moving a robot from a given start vertex to a given goal vertex while the edge costs of a graph change, we have developed D* Lite [16]. It always moves the robot on a shortest path from its current

vertex to the goal vertex, and replans the shortest path when the edge costs change. D* Lite combines ideas from Lifelong Planning A* and Focussed D* [34]. It implements the same navigation strategy as Focussed D* but is simpler and consequently easier to understand and analyze, yet is at least as efficient. (In fact, our experiments show that it is even more efficient than Focussed D*.)

In the following, we first describe a particular mapping paradigm that we call Greedy Mapping. Then we explain how to model it as a graph-search problem and use D* Lite to solve it.

3 Greedy Mapping

Mapping is an important task for mobile robots and a large number of mapping methods have been developed for them, both in robotics and in theoretical computer science [9, 20, 13, 18, 3, 8, 12, 21, 23, 29, 2, 10, 11, 19, 30, 27, 6, 24, 25]. A good overview is given in [31]. In this paper, we study Greedy Mapping, a simple sensor-based planning method that discretizes terrain into cells and then always moves the robot from its current cell to the closest cell with unknown blockage status, that is, to the closest "unobserved cell," until the terrain is mapped. It is greedy because its plans quickly gain information but do not take the long-term consequences of the movements into account. Greedy Mapping is probably one of the first mapping methods that come to mind when empirical robotics researchers quickly need to implement a mapping method, and versions of it have been used on robots by different research groups [36, 17, 32]. It has the following desirable properties:

– **Theoretical Foundation:** Greedy Mapping has a solid theoretical foundation that allows one to characterize its behavior analytically. For example, it is guaranteed to map terrain under realistic assumptions and its plan-execution time can be analyzed formally. In fact, one can prove that its mapping time is reasonably small [17].
– **Simple Integration into Robot Architectures:** Greedy Mapping is simple to implement and integrates well into complete robot architectures. It is robust with respect to the inevitable inaccuracies and malfunctions of other architecture components. For example, Greedy Mapping does not need to have control of the robot at all times. This is important because search methods should only provide advice on how to act and work robustly even if that advice is ignored from time to time [1]. For example, if a robot has to re-charge its batteries during mapping, then it might have to preempt mapping and move to a known power outlet. Once restarted, the robot should be able to resume mapping from the power outlet, instead of having to return to the cell where mapping was stopped (which could be far away) and resume its operation from there. Greedy Mapping exhibits this behavior automatically. Similarly, Greedy Mapping does not expect that the robot faithfully follows its recommendations. Obstacle avoidance, for example, can change

the proposed movement to avoid that the robot gets too close to obstacles. In this case, the robot ends up at a location different from where Greedy Mapping expected it to move. This is not a problem for Greedy Mapping since it automatically resumes its operation from there.
- **Prior Knowledge:** Greedy Mapping takes advantage of prior knowledge about parts of the terrain (if available) since it uses all of its knowledge about the terrain when it determines which unobserved cell is closest to the robot and how to get there quickly. It does not matter whether this knowledge was previously acquired by the robot or provided to it.
- **Distributed Search:** Mapping tasks can be solved with several robots that each run Greedy Mapping and share their maps, thereby decreasing the mapping time. Cooperative mapping is a currently very active research area [7, 33].

Greedy Mapping frequently needs to determine a shortest path from the current cell of the robot to the closest unobserved cell. In the following, we show how D* Lite can be used to implement Greedy Mapping efficiently. This is the first demonstration of how lifelong planning methods can be used to map unknown terrain. The method is to the best of our knowledge, the first efficient implementation of Greedy Mapping to date.

4 Representation as Graph-Search Problem

The mapping problem can be formulated as a graph coverage problem, for example, by imposing a regular eight-connected grid over the terrain. The vertices of the directed graph correspond to cells and are either blocked or unblocked. The robot moves along the edges of the graph but cannot move onto blocked vertices. The robot knows the graph but initially does not know exactly which vertices are blocked. It can utilize initial knowledge about the blockage status of vertices in case it is available. For example, Figure 1 (left) shows a terrain with some prior knowledge about the blockage status of vertices, and Figure 2 (left) shows the corresponding grid. Black cells (vertices) are known to be blocked and white cells are known to be unblocked. Grey cells have an unknown blockage status that the robot needs to determine. The robot has an on-board sensor that reports the blockage status of vertices close to it, including the blockage status of the vertices adjacent to the current vertex of the robot. Greedy Mapping remembers this information and uses it to always move the robot on a shortest potentially unblocked path to the closest unobserved vertex. (A potentially unblocked path is one that does not contain vertices that are known to be blocked.) Figure 1 (left) shows such a path. While following this path, the robot is guaranteed to find out about the blockage status of at least one vertex whose blockage status it did not know about before and is thus guaranteed to make progress with mapping. This is so because it will eventually observe the blockage status of the vertex it navigates to if it does not gain information about the blockage status of other vertices earlier. Figure 1 (right) shows that a robot with a sensor range of two cells observes that cell B5 is blocked after it moves one cell to the east along the planned path. Whenever the robot gains information about the blockage status of vertices, the shortest potentially unblocked path from its current vertex to the closest vertex with unknown blockage status can change, either because the closest vertex with unknown blockage status changes or the path to it changes. Similarly, whenever the robot deviates from the planned path, the shortest potentially unblocked path from its current vertex to the closest vertex with unknown blockage status can change because its current vertex is no longer on the planned path. In both cases, Greedy Mapping needs to recalculate a shortest potentially unblocked path to the closest vertex with unknown blockage status. Figure 1 (right) shows the new path.

5 Solving the Graph Search Problem with D* Lite

Greedy Mapping could be implemented with any graph-search method. In the following we explain the advantages of implementing it with D* Lite rather than other search methods. D* Lite uses two different ways of speeding up its searches. It is a heuristic search method and thus uses heuristic knowledge to focus the search and speed it up. It is also a lifelong planning method and thus uses information from previous searches to speed up the search. In the following, we explain why both heuristic and lifelong searches have the potential to recalculate a shortest potentially unblocked path from the current vertex of the robot to the closest vertex with unknown blockage status faster than other search methods.

D* Lite can use heuristic information to speed up its search, in form of estimates of the distances between two vertices of the graph. The heuristics could estimate the distance from a vertex to the closest unobserved vertex. However, this distance is hard to estimate since there are often several of them in different directions around the robot. Instead, the heuristics of D* Lite estimate the distance from a vertex to the current vertex of the robot. D* Lite can use these heuristics because it effectively performs a backward search from all unobserved vertices to the current vertex of the robot after it has reversed all edges of the graph. These heuristics can easily be obtained since vertices correspond to cells. We calculate an estimate of the distance between two vertices on an eight-connected grid as the maximum of the absolute differences of their x and y coordinates of the corresponding cells. This results in consistent heuristics that are for eight-connected grids what Manhattan distances are for four-connected grids. For example, the heuristic value of vertex B4 is two in Figure 1 (left) but only one in Figure 1 (right).

D* Lite can also use information from previous searches to speed up its search. Figure 1 demonstrates how reusing information can potentially save search effort in the context of Greedy Mapping. The left part of the figure shows the distances of all vertices that are known to be unblocked to the closest cell with unknown blockage status. The right part of the figure shows the same distances after the robot moved one cell to the east along the planned path and gained information about the blockage status of cell B5. All but three distances (shown

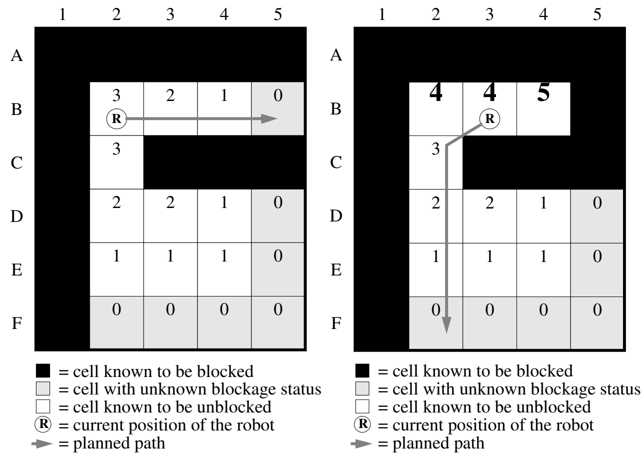

Fig. 1. Example Mapping Task

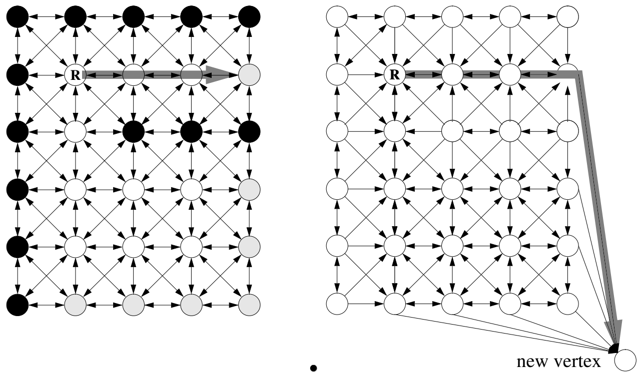

Fig. 2. Graphs for the Example Mapping Task

in bold) remain unchanged and therefore do not need to get recomputed. This suggests that reusing information from previous searches can potentially reduce the search time of heuristic search methods for Greedy Mapping. For example, when replanning the shortest path in Figure 2 (right), D* Lite only expands the three vertices whose distances to the closest cell with unknown blockage status have changed (namely B2, B3, and B4). On the other hand, A* expands seven vertices (namely C2, D2, D4, D5, E2, E5, F2) and thus is less efficient than D* Lite.

6 Technical Details

In the following, we demonstrate how D* Lite can be used to implement Greedy Mapping. We introduce a new vertex and then construct the so-called extended graph as follows: First, the extended graph contains all edges from the graph that corresponds to the terrain (in the following called the original graph), except those edges that go from vertices that are known to be unblocked or potentially unblocked to vertices that are known to be blocked. This ensures that the planned path cannot contain vertices that are known to be blocked and thus is potentially unblocked. (The extended graph contains edges of the original graph that go from vertices that are known to be blocked to other vertices. This is important in case the robot has, due to sensor or position uncertainty, mistakenly classified an unblocked vertex as blocked and then, due to actuator uncertainty, deviates from the planned path and reaches this vertex. The robot believes that it can leave this vertex only if the edges of the extended graph that go from it to other vertices have not been deleted.) Second, the extended graph contains edges that go from any vertex with unknown blockage status that can be reached with one edge traversal from vertices with known blockage status to the new vertex. This ensures that the planned path reaches a vertex with unknown blockage status and, from there, the new vertex. Figure 2 shows the extended graph (right) that corresponds to the graph (left) that models the terrain in Figure 1 (left). A shortest potentially unblocked path on the extended graph from the current vertex of the robot to the new vertex corresponds to a shortest unblocked path on the original graph from the current vertex of the robot to a vertex with unknown blockage status, and vice versa. Thus, Greedy Mapping can determine a shortest potentially traversable path in the original graph from the current vertex of the robot to the closest vertex with unknown blockage status by finding a shortest path in the extended graph from the current vertex of the robot to the new vertex. Thus, D* Lite can be applied to the extended graph. This way it finds the path in the extended graph that is shown in Figure 2 (right). This path corresponds to the path in the original graph that is shown in Figure 2 (left) and the path in the terrain that is shown in Figure 1 (left).

7 Integration into Robot Architectures

We integrated Greedy Mapping into a multi-task autonomous robot architecture called MissionLab [22], which is a version of the Autonomous Robot Architecture (AuRA) [5]. AuRA is a hybrid system that consists of a schema-based reactive system at the low level and a deliberative system based on finite state automata at the high level. The reactive component consists of primitive behaviors called motor schemata [4] that are grouped into behavioral assemblages. Each behavior of a behavioral assemblage produces its own recommendation for how the robot should move, in form of a vector. The robot then moves in the direction of the weighted average of all vectors. We utilize that Greedy Mapping makes only action recommendations and thus can coexist with other components of a robot architecture that also make action recommendations. This allows us to implement map building as a behavioral assemblage of three behaviors that takes as parameters the bounds of the area that the robot needs to map: GreedyMapping directs the robot towards the closest cell with unknown blockage status;

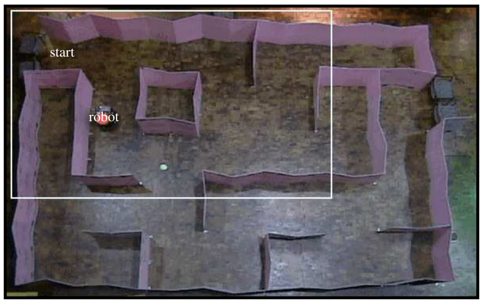

Fig. 3. Maze

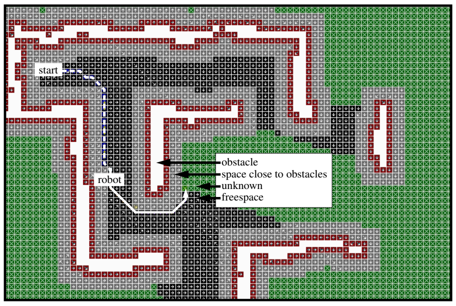

Fig. 4. Screen Shot of Learned Map

AvoidObstacles repels the robot from obstacles; and Wander injects some noise. The weight of AvoidObstacle and the distance within which obstacles affect the robot are set depending on the size of the grid. For very small grids, the weight can be set to zero since Greedy Mapping can directly take the obstacles into account. For larger grids, the weight can be set to a positive value while the distance within which obstacles affect the robot is made small in comparison to the grid size. This ensures that the robot successfully navigates around small obstacles. The weight of Wander is configured similarly.

We used Greedy Mapping both on a Nomad 150 with a Sick LMS200 laser scanner and in simulation in conjunction with a simple 8-connected grid, the cells of which had a size of 10 centimeters by 10 centimeters. All processing was performed on-board the robot on a Toshiba Pentium MMX 233 MHz laptop running Redhat 6.2 Linux. The robot interleaved sensing, planning, and movement. Sensing consisted of a full 180 degree scan with the laser scanner. Initially, all cells of the grid were marked as unobserved. The cells that corresponded to detected obstacles were marked as observed and untraversable. Obstacles were surrounded by traversable cells with a large cost, to bias the robot away from them. The other cells swept by the sensor were marked as observed and traversable. Cells at distance one from obstacles had traversal cost ten, cells at distances two or three from obstacles had traversal cost five, and cells at larger distances from obstacles had traversal cost one. Planning found a shortest path from the current cell of the robot to the closest unobserved cell, the first action of which was executed. Then, the cycle repeated until all cells had been observed or the shortest path to the closest unobserved cell had infinite cost.

We used the robot to map a maze of size 28 by 20 feet that we constructed out of polystyrene insulation on the ground floor of our building. We let the robot map the maze five times. All five experiments were successful. Figure 3 shows a top view of the maze. Figure 4 shows a snapshot of the map during map building, together with the shortest path from the current cell of the robot to a closest unobserved cell. The part of the maze that corresponds to the part of the map shown in the screen shot is outlined in Figure 3.

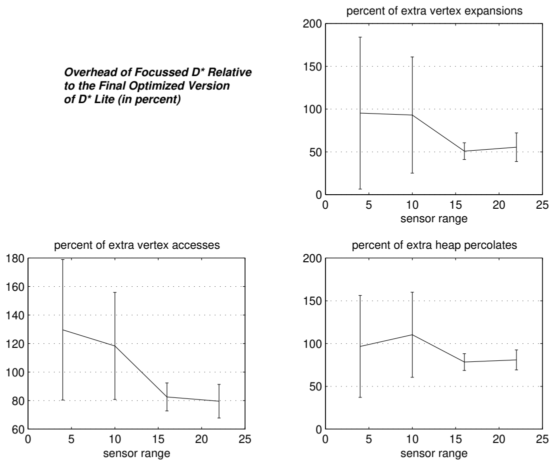

Fig. 5. Comparison of D* Lite and D*

The version of Greedy Mapping used in this paper assumes that there is neither position nor sensor uncertainty. The assumption that there is no position uncertainty makes Greedy Mapping well suited for outdoor navigation in conjunction with GPS. This assumption was not justified in our experiments since the cell of the robot was estimated using a simple dead-reckoning technique. The map shown in Figure 4 shows some of the resulting inaccuracies. However, the runs were not long enough for this to become a problem. The assumption that there is no sensor uncertainty was justified. The Sick laser scanner is highly accurate and has sufficient resolution and range accuracy.

8 Experimental Evaluation

In the previous section we demonstrated that Greedy Mapping is easy to implement and integrate into complete robot architectures. We now want to demonstrate the advantage of D* Lite over other search methods for implementing Greedy Mapping. We thus compare D* Lite against D*. We also compare it against D* Lite without heuristic search, D* Lite without lifelong planning (that is, an A* search toward the current cell of the robot), and D* Lite without both heuristic and lifelong search (that is, a breadth-first search toward the current cell of the robot). Since all search methods move the robot in the same way, we only need to compare their total planning time. Since the actual planning times are implementation-dependent, we instead use three performance measures that all correspond to common operations performed by D* Lite and thus heavily influence its planning time: the total number of vertex expansions (vertices that are expanded more than once increase the number of vertex expansions appropriately), the total number of heap percolates (exchanges of a parent and child

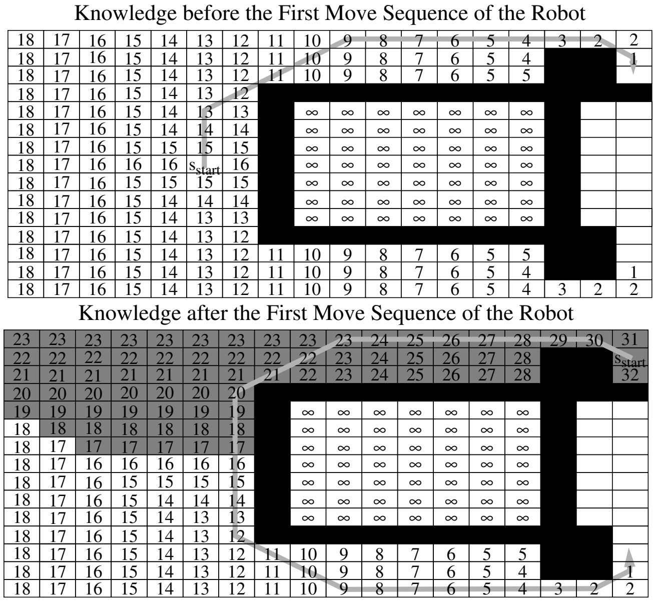

Fig. 6. Illustration of the Behavior of Different Versions of D* Lite (1)

in the heap), and the total number of vertex accesses (for example, to read or change their values).

- We first study to which degree D* Lite outperforms D*. We perform experiments in 25 randomly generated terrains of size 64 by 25 that are represented as eight-connected grids and resemble office environments. The robot has no prior knowledge of the terrain. We use the MissionLab simulation to be able to average the results over several runs. We varied the sensor range of the robot to simulate both short-range and long-range sensors. For example, if the sensor range is four, then the robot can sense all blocked cells that are up to four cells in any direction away from the robot as long as they are not blocked from view by other blocked cells. Figure 5 shows the three performance measures for Greedy Mapping with D* as percent difference relative to Greedy Mapping with D* Lite. (Thus, D* Lite always scores zero.) The figure reports not only the means of the three performance measures but also the corresponding 95 percent confidence intervals to demonstrate that our conclusions are statistically significant. D* Lite performs better than D* with respect to all three measures, justifying our claim that it is more efficient than Focussed D*.
- We now study to which degree the combination of lifelong-planning and heuristic search that D* Lite implements outperforms lifelong planning or heuristic searches individually. Figure 6 shows an example for sensor rage one, where the robot has some prior knowledge of the terrain, both before and after the robot has moved along the path and discovered the blockage

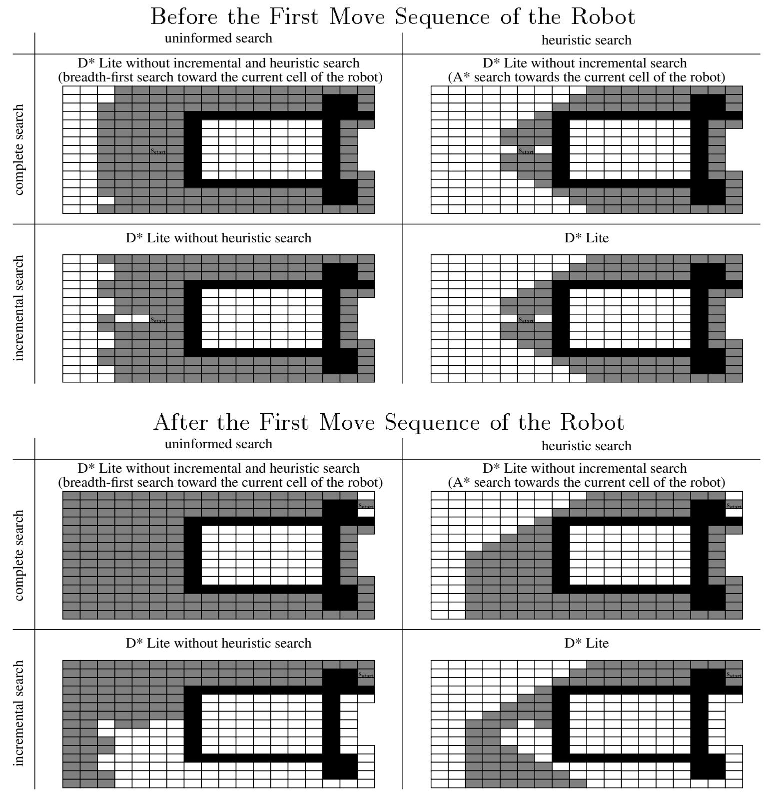

Fig. 7. Illustration of the Behavior of Different Versions of D* Lite (2)

status of an additional cell. Cells are empty if their blockage status is unknown. The arrows show a shortest path from the current cell of the robot to a closest cell with unknown blockage status. Cells known to be traversable are labeled with the distance to the closest cell with unknown blockage status. Cells whose distances have changed are shaded gray. Notice that the number of cells with changed distances is small and many of the changed distances are irrelevant for recalculating the path. Figure 7 shows the expanded cells by the different versions of D* Lite in gray. As the figures show, the heuristic search outperforms the uninformed search, and the incremental search outperforms the complete (that is, nonincremental) one after the first search (where previous search results are available). The figures also illustrate that the combination of heuristic and incremental search performed by D* Lite decreases the number of expanded cells even more than either a heuristic search or an incremental search individually. In particular, the first search of D* Lite expands exactly the same cells as an A* search if A* breaks ties between vertices with the same f-values suitably. The small differences are due only to the different tie-breaking behavior. The second

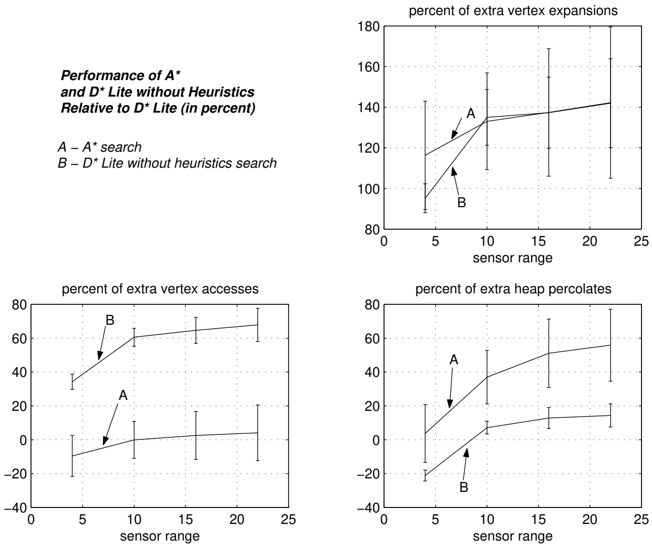

Fig. 8. Comparison of Different Versions of D* Lite

search of D* Lite expands only a subset of those cells whose goal distances changed (or had not been calculated before). Thus, D* Lite results in substantial savings over an A* search that replicates most of its previous search. To test whether these results are statistically significant, we perform again experiments in 25 randomly generated terrains of size 64 by 25 that are represented as eight-connected grids and resemble office environments. The robot has no prior knowledge of the terrain. Figure 8 shows the three performance measures for Greedy Mapping without heuristic search and D* Lite without lifelong planning as percent difference relative to Greedy Mapping with D* Lite. We decided not to include D* Lite without both heuristic and lifelong search because it performs so poorly that graphing its performance becomes a problem. As can be seen, the number of vertex expansions of D* Lite is always much smaller than that of the other two algorithms. This also holds for the number of heap percolates and vertex accesses, with the exception of sensor range four for the heap percolates. The advantage of D* Lite over the other two search methods seems to increase as the sensor range increases. This is important since laser scanners tend to be the sensors of choice for mapping. Only for the number of vertex accesses is the difference between D* Lite and D* Lite without lifelong planning search statistically not significant although, for the larger sensor ranges, the mean of D* Lite is smaller than that of D* Lite without lifelong planning.

9 Conclusions

In this paper, we have explained how lifelong planning applies to mapping of unknown terrain, a new application of lifelong planning. Our results show that the combination of lifelong and heuristic search that D* Lite implements speeds up the planning time of Greedy Mapping over lifelong or heuristic searches individually. These results demonstrate the versatility and computational benefits of lifelong planning, an underexploited technique that should be used more often in mobile robotics.

Acknowledgments

We thank William Halliburton for interfacing D* Lite to the hardware and performing the robot experiments. We thank Anthony Stentz for his support. The Intelligent Decision-Making Group is partly supported by NSF awards under contracts IIS-9984827, IIS-0098807, and ITR/AP-0113881. 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 and agencies or the U.S. government.

References

1. P. Agre and D. Chapman. Pengi: An implementation of a theory of activity. In *Proceedings of the National Conference on Artificial Intelligence*, pages 268–271, 1987.
2. S. Albers and M. Henzinger. Exploring unknown environments. *SIAM Journal on Computing*, 29(4):1164–1188, 2000.
3. R. De Almeida and C. Melin. Exploration of unknown environments by a mobile robot. *Intelligent Autonomous Systems*, 2:715–725, 1989.
4. R. Arkin. Motor-schema based mobile robot navigation. *International Journal of Robotics Research*, 8(4):92–112, 1989.
5. R. Arkin and T. Balch. AuRA: Principles and practice in review. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2):175–189, 1997.
6. B. Awerbuch, M. Betke, R. Rivest, and M. Singh. Piecemeal graph exploration by a mobile robot. *Information and Computation*, 152(2):155 –172, 1999.
7. W. Burgard, D. Fox, M. Moors, R. Simmons, and S. Thrun. Collaborative multi-robot exploration. In *Proceedings of the International Conference on Robotics and Automation*, pages 476–481, 2000.
8. C. Choo, J. Smith, and N. Nasrabadi. An efficient terrain acquisition algorithm for a mobile robot. In *Proceedings of the International Conference on Robotics and Automation*, pages 306–311, 1991.
9. H. Choset. *Sensor-Based Motion Planning: The Hierarchical Generalized Voronoi Graph.* PhD thesis, California Institute of Technology, Pasadena (California), 1996.
10. X. Deng, T. Kameda, and C. Papadimitriou. How to learn an unknown environment; I: The rectilinear case. *Journal of the ACM*, 45(2):215 –245, 1998.
11. X. Deng and C. Papadimitriou. Exploring an unknown graph. In *Proceedings of the Symposium on Foundations of Computer Science*, pages 355–361, 1990.

12. G. Dudek, P. Freedman, and S. Hadjres. Using local information in a non-local way for mapping graph-like worlds. In *Proceedings of the International Conference on Artificial Intelligence*, pages 1639–1647, 1993.
13. G. Dudek, M. Jenkin, E. Milios, and D. Wilkes. Robotic exploration as graph construction. *IEEE Transactions on Robotics and Automation*, 7(6):859–865, 1991.
14. D. Frigioni, A. Marchetti-Spaccamela, and U. Nanni. Fully dynamic algorithms for maintaining shortest paths trees. *Journal of Algorithms*, 34(2):251–281, 2000.
15. S. Koenig and M. Likhachev. Incremental A*. In *Proceedings of the Neural Information Processing Systems*, 2002. (to appear).
16. S. Koenig and M. Likhachev. Improved fast replanning for goal-directed navigation in unknown terrain. (submitted).
17. S. Koenig, C. Tovey, and W. Halliburton. Greedy mapping of terrain. In *Proceedings of the International Conference on Robotics and Automation*, pages 3594–3599, 2001.
18. B. Kuipers and Y. Byun. A robust, qualitative method for robot spatial learning. In *Proceedings of the National Conference on Artificial Intelligence*, pages 774–779, 1988.
19. S. Kwek. On a simple depth-first search strategy for exploring unknown graphs. In *Proceedings of the Workshop on Algorithms and Data Structures*, volume 1272 of *Lecture Notes in Computer Science*, pages 345 –353. Springer, 1997.
20. J. Leonard, H. Durrant-Whyte, and I. Cox. Dynamic map building for an autonomous mobile robot. *International Journal of Robotics Research*, 11(4):286–298, 1992.
21. V. Lumelsky and A. Stepanov. Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, 2:403–430, 1987.
22. D. Mackenzie, R. Arkin, and J. Cameron. Multiagent mission specification and execution. *Autonomous Robots*, 4(1):29–57, 1997.
23. G. Oriolo, G. Ulivi, and M. Vendittelli. Real-time map building and navigation for autonomous robots in unknown environments. *IEEE Transactions on Systems, Man, and Cybernetics*, 28(3):316–333, 1998.
24. P. Panaite and A. Pelc. Exploring unknown undirected graphs. *Journal of Algorithms*, 33(2):281 –295, 1999.
25. C. Papadimitriou and M. Yannakakis. Shortest paths without a map. *Theoretical Computer Science*, 84(1):127–150, 1991.
26. J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Addison-Wesley, 1985.
27. L. Prasad and S. Iyengar. A note on the combinatorial structure of the visibility graph in simple polygons. *Theoretical Computer Science*, 140(2):249–263, 1995.
28. G. Ramalingam and T. Reps. An incremental algorithm for a generalization of the shortest-path problem. *Journal of Algorithms*, 21:267–305, 1996.
29. N. Rao. Algorithmic framework for learned robot navigation in unknown terrains. *IEEE Computer*, 22(6):37–43, 1989.
30. N. Rao. Robot navigation in unknown generalized polygonal terrains using vision sensors. *IEEE Transactions on Systems, Man, and Cybernetics*, 25(6):947–962, 1995.
31. N. Rao, S. Hareti, W. Shi, and S. Iyengar. Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms. Technical Report ORNL/TM–12410, Oak Ridge National Laboratory, Oak Ridge (Tennessee), 1993.

32. L. Romero, E. Morales, and E. Sucar. An exploration and navigation approach for indoor mobile robots considering sensor's perceptual limitations. In *Proceedings of the International Conference on Robotics and Automation*, pages 3092–3097, 2001.
33. K. Singh and K. Fujimura. Map making by cooperating mobile robots. In *Proceedings of the International Conference on Robotics and Automation*, pages 254–259, 1993.
34. A. Stentz. The focussed D* algorithm for real-time replanning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1652–1659, 1995.
35. S. Thrun. Lifelong learning algorithms. In S. Thrun and L. Pratt, editors, *Learning To Learn*. Kluwer Academic Publishers, 1998.
36. S. Thrun, A. Bücken, W. Burgard, D. Fox, T. Fröhlinghaus, D. Hennig, T. Hofmann, M. Krell, and T. Schmidt. Map learning and high-speed navigation in RHINO. In D. Kortenkamp, R. Bonasso, and R. Murphy, editors, *Artificial Intelligence Based Mobile Robotics: Case Studies of Successful Robot Systems*, pages 21–52. MIT Press, 1998.