RC25217 (W1110-056) October 11, 2011
Computer Science

IBM Research Report

Path Planning Problems Motivated by a Data Center Robot

Ning Xu
City College of New York

Jonathan Lenchner
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Cambridge - Haifa - India - T. J. Watson - Tokyo - Zurich

Path Planning Problems Motivated by a Data Center Robot

Ning Xu* Jonathan Lenchner†

1 Introduction

A robot is responsible for navigating, creating a map of, and then monitoring a computer data center for temperature and other environmental parameters. The floor of a data center is comprised of industry-standard $2' \times 2'$ tiles. Some tiles are **occupied** by obstacles (e.g. equipment, walls) such that the robot cannot enter. The other tiles are **empty**. Two tiles are **connected** if they share a common side – in other words, the robot can only move orthogonally to the tile boundaries. All empty tiles are assumed to be connected. Figure 1 illustrates the map of a data center. In one model the robot is equipped with a camera to view

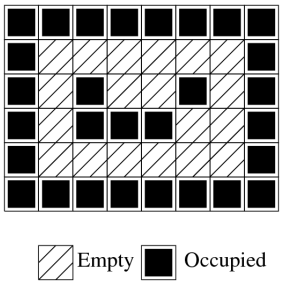

Figure 1: A data center – the black tiles are occupied, the other tiles are empty.

the tile in the direction it is facing. In another model, the robot is equipped with three cameras to view the tile in front of it as well as the tiles on its two sides. By viewing a tile, the robot can verify whether the tile is empty or occupied, and maintain an evolving map of the explored area of the data center in its memory. The development of, and experiences with, such robots are described in [3].

In the course of its navigation, the robot can either move one tile forward, that is, move to the adjacent tile in its facing direction, if such a tile is empty, or turn by 90 degrees, that is, change its facing direction to one of its side directions.

Let 1 be the time cost of a move, and τ be the time cost of a turn. It is an important design principle that the robot find a closed tour with minimal or near-minimal time cost, such that the robot visits all empty tiles and returns to its starting tile. We call this problem the **data center navigation problem**. We distinguish two variants of the navigation problem. When the map of the data center is unknown in advance, we call the problem **the data center exploration problem**. When the map is known before exploration, we call the problem **the data center monitoring problem**. In our ensuing analysis we take n to be the number of visitable tiles in the data center.

*City College of New York
†IBM T.J. Watson Research Center

2 Complexity

Arkin et al. [1] showed that the data center monitoring problem is NP-hard, even when $\tau = 1$ (the value of τ which best models the robot in [3]). The proof proceeded by first showing that deciding the Hamiltonicity of axis-parallel unit segment intersection graphs (HUSIG) is NP-hard and then reducing HUSIG to the data center monitoring problem. Since the data center exploration problem is clearly as hard as the data center monitoring problem, it too is NP-hard.

3 The data center exploration problem

There are two intuitive algorithms for the data center exploration problem: a **Greedy** algorithm [4], and an algorithm based on Depth First Search (**DFS**). In either algorithm the robot travels to unvisited, adjacent, empty tiles with minimum cost and when stuck at a tile v, the robot tracks back – to the known-to-be-closest unknown tile in the case of Greedy, or to the unknown tile adjacent to the most recently visited tile in the case of DFS. The greedy algorithm is sometimes referred to as the A* algorithm in artificial intelligence applications.

3.1 Performance of DFS

If the robot is equipped with three cameras, the robot finds all adjacent tiles to a tile v when the robot arrives at v the first time (with the exception of the very first tile). A simple analysis shows that the robot can be required to turn at most twice at each tile. Thus the time cost for all turns is at most $2n\tau$. Therefore, the robot equipped with three cameras can guarantee a $2+2\tau$ factor approximation running DFS.

When the robot is equipped with just one forward-facing camera, the robot can just see the tile it is facing. This means that the robot requires more turns to find adjacent tiles. In this case the robot can be seen to require at most four turns in each tile, so the time cost of turns is at most $4\tau n$. The time cost of moves is still $2n - 2$.

Therefore, for the robot with only one camera, the total time cost is at most $2n - 2 + 4\tau n$, i.e., the DFS algorithm can guarantee a $2 + 4\tau$ factor approximation.

3.2 Performance of Greedy

Unlike DFS, the greedy algorithm cannot guarantee a constant factor approximation.

Theorem 1. *For any constant $\delta > 0$, there are instances of the Exploration problem in which Greedy does not complete its exploration with time cost $\delta \times OPT$, where OPT is the optimal time cost.*

The recursive construction that provides the proof of this theorem is quite involved. In practice it is difficult to even come up with examples where Greedy does *any* worse than DFS. Figure 2 gives such an example using

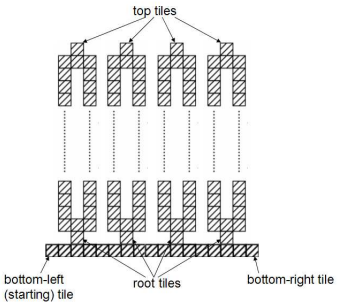

Figure 2: An example where DFS outperforms Greedy.

one of the fundamental constructs in the proof of Theorem 1. If the robot starts at the bottom-left tile in the figure it will go all the way to the right-most bottom tile, then come up to the right-most root tile, traverse around the associated loop, not stopping at the associated top tile (since at that point it is cheaper to go forward to find an unvisited tile then turn and go forward to get to an unvisited tile). By the time it returns to the vicinity of the root tile, it will be closer to go to the next closest root tile, rather than handling the left-over top tile which it has left "stranded." The robot proceeds from right to left until just the top tiles remain, and then must return to visit these. Had it been using DFS instead of Greedy the robot would have returned to visit the respective top tiles, before proceeding back to the prior root tiles. This example shows that DFS can be just a small constant factor better than Greedy. A multi-level version of this construction shows that Greedy can be an arbitrarily bad constant factor approximation compared to DFS and hence cannot provide a fixed constant factor approximation to OPT.

4 Approximating the data center monitoring problem

The data center monitoring problem is NP-hard, as described in the Section 2. One can apply a minor variation of either DFS or Greedy as discussed in the previous section to this problem as well. In the case of the monitoring problem there is no advantage to having three cameras and it is easy to see that DFS for one or more cameras achieves a $2 + 2\tau$ approximation ratio using $O(n)$ time and $O(n)$ space.

The asymptotically worst run-time can be reduced by adapting the Christofides approximation to TSP [2]. One first uses Christofides to get an approximately optimal tour on the all-pairs shortest path graph without turn costs, and then looks back to make some simple observations about the maximum number of turns incurred when taking such a tour compared with OPT.

Theorem 2. *The adapted Christofides algorithm can monitor the data center with run-time cost which is at most $1.5 + 2\tau$ times the optimal solution, using $O(n^3)$ time and $O(n^2 \log n)$ space.*

5 A practical refinement

In real data center layouts we have found that Greedy almost always does better than DFS or the adapted Christofides algorithm in the data center monitoring task. There is a refinement to Greedy that warrants mentioning. Rather than going to the closest unvisited tile, instead compute the shortest travel times to all unvisited tiles and divide each such number by the number of unvisited tiles encountered along the way to obtain an amortized cost and minimize over all such amortized costs. This not-quite-so-greedy algorithm again runs in time $O(n)$, but requires $O(n \log n)$ space.

References

[1] Arkin, E.M., Bender, M.A., Demaine, E.D., Fekete, S.P., Mitchell, J.S.B., Sethia, S., 2001, Optimal covering tours with turn costs, Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms, 138-147.

[2] N. Christofides, 1976, Worst-case analysis of a new heuristic for the travelling salesman problem, Report 388, Graduate School of Industrial Administration, CMU.

[3] Lenchner, J., Isci, C., Kephart, J.O., Mansley, C., Connell, J., McIntosh S., 2011, Towards Data Center Self-Diagnosis Using a Mobile Robot, Proceedings of IEEE International Conference of Autonomic Computing, Karlsruhe, Germany.

[4] Panaite, P. and Pelc, A., 1999, Exploring Unknown Undirected Graphs, Journal of Algorithms, 33, 281-295.