Efficient and Complete Centralized Multi-Robot Path Planning

Ryan Luna and **Kostas E. Bekris**
University of Nevada, Reno
1664 N. Virginia St., MS 171
Reno, NV 89557 USA
{rluna, bekris}@cse.unr.edu

Introduction

Multi-robot path planning requires the computation of a set of compatible paths for multiple robots operating on a discrete roadmap. The goal of this problem is to navigate each robot to their unique goal vertices without simultaneously occupying the same vertex or edge in the roadmap as any other robot. Such a formulation has applications in warehouse management, transportation networks, (dis)assembly, robotic mining, space exploration, and computer games.

The problem of multi-robot path planning has been extensively studied in the literature, with coupled techniques that attempt to intelligently prune the search space into something more tractable while maintaining completeness, as well as decoupled techniques that plan for each robot independently and utilize sophisticated heuristics in order to avoid collisions along these paths.

Coupled search techniques have been employed to separate large problems into fully coupled sub-problems (van den Berg et al. 2009), or segment the roadmap into smaller topologies with known characteristics (Ryan 2008) to solve the composite problem sequentially.

Decoupled techniques employ heuristics that prioritize robots (Erdmann and Lozano-Perez 1986), or tune the velocities on precomputed paths (Kant and Zucker 1986). Modern approaches consider dynamic prioritization and windowed search (Silver 2005), or domain restriction to guarantee completeness and tractability (Wang and Botea 2009).

Contribution

This work provides a novel, algorithmically complete approach for multi-robot path planning that efficiently computes sequential solutions. The algorithm is complete for a general class of problems: instances in which there are at most $n - 2$ robots on a roadmap of n vertices. Experimental results show that the technique is able to compute solutions many orders of magnitude faster than a traditional A^* implementation while maintaining completeness, and in times highly competitive with a state-of-the-art decoupled and centralized planners. The approach makes no assumption about the topology of the roadmap, and does not depend on tunable parameters to solve a problem effectively.

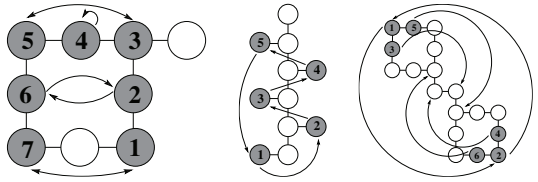

Figure 1: Small multi-robot path planning instances difficult for coupled and decoupled techniques. From left: Loopchain, string, connector.

Push and Swap Approach

The proposed PUSH_AND_SWAP technique utilizes a set of simple and computationally cheap primitives, PUSH and SWAP, to solve multi-robot path planning problems in a sequential manner. As shown in Algorithm 1, the proposed method simply iterates over the set of robots and first *pushes* the robot towards its goal, given a set of positions that must remain static. If the robot a is not able to push to its goal, it must *swap* positions with the adjacent robot along its shortest path. This loop of *pushing* and *swapping* continues until robot a is at its goal. Once this occurs, the goal is added into the list of static positions (line 7), and the loop repeats for the next robot. Algorithms for the PUSH and SWAP primitives are omitted for brevity, but are briefly discussed below.

Algorithm 1 PUSH_AND_SWAP($\mathcal{A}, \mathcal{S}, \mathcal{T}$)

```
1: U ← ∅
2: for all a ∈ A do
3:     while S[a] ≠ T[a] do
4:         if PUSH(a, U) == FALSE then
5:             if SWAP(a) == FALSE then
6:                 return FALSE (i.e., Failure)
7:     U ← U ∪ T[r]
8: return TRUE (i.e., Success)
```

Push primitive: The push primitive navigates a robot along its shortest path, potentially detouring other robots in the process. In short, when a robot is blocking the shortest path of the pushing robot a, PUSH attempts to move the robot blocking a away from its current position, allowing a to advance along its shortest path. There are restrictions, however, on what can be "pushed". The set $\mathcal{U}$ contains vertices of robots that have already planned and are at their goal position; these robots should not be moved. If robot a does

Problem	WHCA*(5)	Sp. Tree	Push-and-Swap
Loop-Chain	∞	n/a	8.45
String	1.46	1.97	0.423
Connector	∞	n/a	2.63

Table 1: Computation time in milliseconds for the benchmarks in Figure 1. ∞ represents no solution computed.

not reach its goal because it cannot make progress by "pushing", it returns false, indicating a SWAP is necessary.

Swap primitive: The swap primitive switches the position of a robot a with a robot b that is adjacent to it along the shortest path. One requirement for SWAP is that after execution, the only robots that have changed positions are a and b. This is necessary to ensure completeness. Maintaining this constraint is feasible by allowing all robots to move during the swap, and reversing the actions taken once the swap is complete. In order to maintain the swap, robot a will reverse using b's paths and vice-versa.

Completeness

PUSH_AND_SWAP provides completeness for problems in which there are at least two empty vertices in the roadmap. The theorem can be proven with the following lemmas:
1) The path planning instance is solvable if and only if SWAP can transfer two robots to the vicinity of a vertex v with a degree at least 3 together with two empty vertices.
2) After each iteration of PUSH and SWAP, at least one robot will make progress along the shortest path to its goal.
3) The PUSH and SWAP operators do not disturb robots already at their goal vertices.
4) If the initial configuration is solvable, any permutation achieved using PUSH and SWAP operators will be solvable.

Results

The proposed technique was evaluated with a set of small benchmark problems, Figure 1, where it is possible to compare against a complete, centralized A^* implementation, a modern decoupled technique, WHCA* (Silver 2005), and a scalable centralized planner that utilizes the spanning tree of the roadmap (Peasgood, Clark, and McPhee 2008). Table 1 shows computation times for the planners on the benchmarks. The spanning tree planner is only applicable in trees where the number of leaves is greater than the number of robots. Times longer than 60 minutes are deemed a failure.

Large scale experiments in a 20x30 grid environment with varying numbers of robots are also performed. The coupled A^* approach, even with "optimal decoupling" applied (van den Berg et al. 2009), still exhibits highly exponential complexity. WHCA* scales well in this environment but suffers from deadlocks as the number of robots increases. The spanning tree planner achieves highly competitive times against the proposed technique, but the path quality achieved by this planner is poor when compared to Push-and-Swap. Figure 2 shows a graph of the Push-and-Swap technique against the other planners. It should be noted that Push-and-Swap found solutions in 100% of simulated experiments.

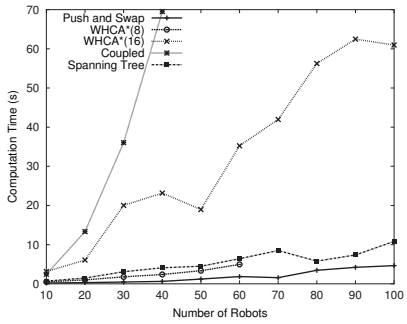

Figure 2: Computation time for robots placed randomly in a 20x30 grid. Values are averages of 20 runs.

Discussion

It is important to note that the solutions generated by PUSH_AND_SWAP are not optimal, and investigating whether an optimal set of paths can be computed using the two primitives is of particular interest. Depending on the problem domain, techniques that provide Pareto optimality (Ghrist, O'Kane, and LaValle 2004) may also be applicable.

In addition to optimality, providing completeness in all solvable sequential instances is desirable. PUSH_AND_SWAP provides completeness in problems where there are at least two empty vertices in the roadmap. The "15-puzzle" problem can easily be formulated as multi-robot path planning, however this instance would only have one empty vertex. It may be possible to create a variant of SWAP to take advantage of redundant loops in order to solve such instances.

References

Erdmann, M., and Lozano-Perez, T. 1986. On multiple moving objects. In *IEEE Intern. Conference on Robotics and Automation (ICRA)*, 1419–1424.

Ghrist, R.; O'Kane, J. M.; and LaValle, S. M. 2004. Pareto optimal coordination on roadmaps. In *Workshop on the Algorithmic Foundations of Robotics (WAFR)*.

Kant, K., and Zucker, S. 1986. Towards efficient trajectory planning: The path-velocity decomposition. *International Journal of Robotics Research (IJRR)* 5(3):72–89.

Peasgood, M.; Clark, C.; and McPhee, J. 2008. A complete and scalable strategy for coordinating multiple robots within roadmaps. *IEEE Transactions on Robotics* 24(2):282–292.

Ryan, M. R. K. 2008. Exploiting subgraph structure in multi-robot path planning. *Journal of Artificial Intelligence Research* 31:497–542.

Silver, D. 2005. Cooperative pathfinding. In *The 1st Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE'05)*, 23–28.

van den Berg, J.; Snoeyink, J.; Lin, M.; and Manocha, D. 2009. Centralized path planning for multiple robots: Optimal decoupling into sequential plans. In *Robotics: Science and Systems V*.

Wang, K.-H. C., and Botea, A. 2009. Tractable Multi-Agent Path Planning on Grid Maps. In *International Joint Conference on Artificial Intelligence IJCAI-09*, 1870–1875.