Late-Breaking Developments in the Field of Artificial Intelligence
Papers Presented at the Twenty-Seventh AAAI Conference on Artificial Intelligence

Fast, Near-Optimal Computation for Multi-Robot Path Planning on Graphs*

Jingjin Yu
Coordinated Science Lab
University of Illinois, Urbana, IL 61801
jyu18@uiuc.edu

Steven M. LaValle
Department of Computer Science
University of Illinois, Urbana, IL 61801
lavalle@uiuc.edu

Abstract

We report a new method for computing near optimal makespan solutions to multi-robot path planning problem on graphs. Our focus here is with hard instances - those with up to 85% of all graph nodes occupied by robots. Our method yields 100-1000x speedup compared with existing methods. At the same time, our solutions have much smaller and often optimal makespans.

Introduction and Problem Formulation

In this paper, we study centralized multi-robot path planning problems on graphs, also known as cooperative path-finding (Silver 2005; Ryan 2008; Standley and Korf 2011; Surynek 2012b). Our focus is on finding plans with optimal or near optimal makespans for problems in which the graph nodes are heavily populated with robots. This problem finds many direct applications in AI and robotics, including microfludics (Ding, Chakrabarty, and Fair 2001) and warehouse automation (Wurman, D'Andrea, and Mountz 2008).

Let $G = (V, E)$ be a connected, undirected, simple graph with vertex set $V = \{v_i\}$ and edge set $E = \{(v_i, v_j)\}$. Let $R = \{r_1, \ldots, r_n\}$ be a set of robots that move with unit speeds along the edges of G, with initial and goal locations on G given by the injective maps $x_I, x_G : R \to V$, respectively. A *path* is a map $p_i : \mathbb{Z}^+ \to V$. A path p_i is *feasible* for a robot r_i if it satisfies the following properties: (1) $p_i(0) = x_I(r_i)$, (2) for each i, there exists a smallest $k_i^{\min} \in \mathbb{Z}^+$ such that for all $k \geq k_i^{\min}$, $p_i(k) \equiv x_G(r_i)$, and (3) for any $0 \leq k < k_i^{\min}$, $(p_i(k), p_i(k+1)) \in E$ or $p_i(k) = p_i(k+1)$. We say that two paths p_i, p_j are in *collision* if there exists $k \in \mathbb{Z}^+$ such that $p_i(k) = p_j(k)$ (collision on a vertex, or *meet*) or $(p_i(k), p_i(k+1)) = (p_j(k+1), p_j(k))$ (collision on an edge, or *head-on*). If $p(k) = p(k+1)$, then the robot stays at vertex $p(k)$ between the time steps k and $k+1$.

Problem (MPP_{pr}). Given (G, R, x_I, x_G), find a set of paths $P = \{p_1, \ldots, p_n\}$ such that p_i's are feasible paths for respective robots r_i's and no two paths p_i, p_j are in collision.

*This work was supported in part by NSF grant 0904501 (IIS Robotics), NSF grant 1035345 (Cyberphysical Systems), MURI/ONR grant N00014-09-1-1052, and AFOSR grant FA9550-12-1-0193.

Note that MPP_{pr} ("*pr*" stands for parallel and rotation, respectively) allows rotations of robots along fully occupied cycles, which differs from the pebble motion problem (PMG for short) studied in (Kornhauser, Miller, and Spirakis 1984) and its parallel extension (MPP_{p} for short) studied in (Silver 2005; Surynek 2012b). Since robots capable of moving in parallel should also be able to move along fully occupied cycles, MPP_{pr} is a more natural model than MPP_{p}.

A natural criterion for measuring path set optimality is the number of time steps until the last robot reaches its goal. This is sometimes called the *makespan*, which can be computed from $\{k_i^{\min}\}$ for a feasible path set P as $T_P = \max_{1 \leq i \leq n} k_i^{\min}$. Our goal in this paper is to compute solutions to MPP_{pr} and MPP_{p} that minimizes the makespan T_P.

Our Method

In (Yu and LaValle 2013a), a complete algorithm was given for solving MPP_{pr} with minimum makespan using multi-flow. Let us denote the integer linear programming (ILP) based algorithm as BASEILP. Since the problem is NP-Hard (Yu and LaValle 2013b), fast computational method for solving the minimum makespan MPP_{pr} must resort to heuristics and approximations. We provide such a heuristic here, augmented by additional heuristics obtained via exploring the structure of the ILP model.

The k-splitting heuristic. Our key heuristic uses the simple idea of divide-and-conquer, with the goal of obtaining smaller MPP_{pr} instances that are then fed to the BASEILP algorithm. Due to limited space, we only provide a high level description of the heuristic here, using $k = 2$.

For each robot r_i, given its start vertex, $x_I(r_i)$, and its goal vertex, $x_G(r_i)$, we compute all vertices of G such that the resulting set of vertices are of equal distance to $x_I(r_i)$ and $x_G(r_i)$ or differ in distance by at most one (i.e., such a vertex m_i satisfies the property $|dist(x_I(r_i), m_i) - dist(x_G(r_i), m_i)| \leq 1$), ignoring the presence of other robots. Let this set of vertices be M_i. A member of M_i is then picked as a *waypoint* for robot r_i. Denote the set of waypoints as $\{w_1, \ldots, w_n\}$. This splits the original problem into two subproblems. One of the problem has $x_I(R)$ ($\{w_i\}$) as the start (goal) configuration and the other problem has $\{w_i\}$ ($x_G(R)$) as the start (goal) configuration. It is possible that $M_i \subseteq \{w_1, \ldots, w_{i-1}\}$, i.e., the "best" waypoints for r_i are already used by a robot. In this case, M_i is

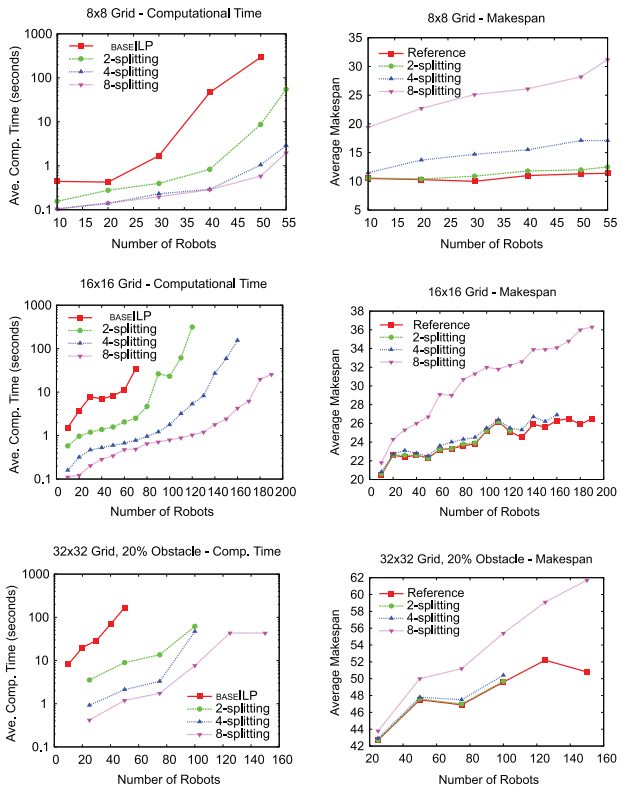

Figure 1: Computational result for MPP_{pr}..

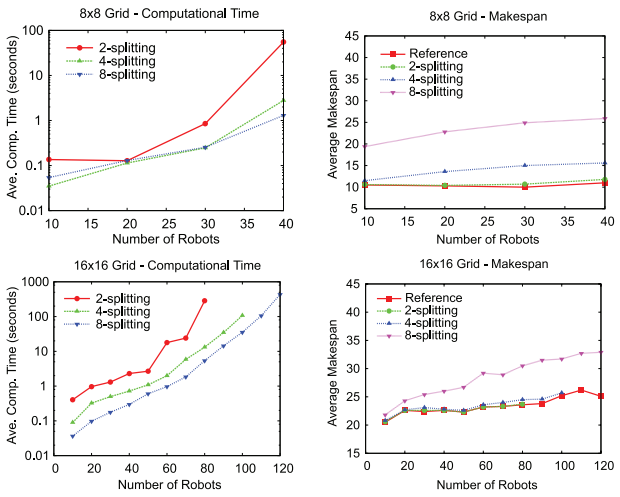

Figure 2: Computational result for MPP_{p}.

expanded incrementally to include additional vertices such that $|dist(x_I(r_i), m_i) - dist(x_G(r_i), m_i)| \leq 2, 3, \ldots$, until a waypoint for r_i becomes available.

Additional heuristics. To get the most performance out of a solver, it is beneficial to have a leaner model (i.e., fewer columns and rows). When it comes to translating the models for an ILP solver, some of the variables can be removed. In our implementation, reachability analysis was used to eliminate (binary) variables that must remain zero. We omit the details of these heuristics due to limited space.

Preliminary Computational Results

We implemented our algorithm in Java and used Gurobi 5.1 (Gurobi Optimization 2012) as our solver. The experiments were carried on an Intel 3970K PC using a 16GB javaVM. As mentioned, our focus is with the hard cases - problems with up to 85% of all vertices occupied by robots. For these problems, the only viable solution appears to be the iCOBOPT method from (Surynek 2012b), which is only designed to solve MPP_{p} prbolems suboptimally. Our method is designed for MPP_{pr} but can also be used for MPP_{p}.

MPP_{pr}. The computation result on MPP_{pr} is given in Fig. 1. Three graph classes are used for this purpose: (1) 8×8 grids with no obstacles, (2) 16×16 grids with no obstacles, and (3) 32×32 grids with 20% of vertices removed to simulate obstacles. Our goal here is to test the effectiveness of the k-splitting huristic (all other heuristics are active in all runs). The start and goal locations are uniformly randomly picked from all available vertices. For each graph/agent combination, ten sequentially randomly generated problems were used for testing. A data point is included only if all ten problems are solved within 1000 seconds. Note that BASEILP always produces true optimal solutions. In the makespan plots, the reference makespan line is computed by treating each robot as the only robot on the graph. Thus, this line is an underestimate of the true optimal makespan.

We observe that the running time decreases rapidly as k is increased in the k-splitting heuristic. Moreover, by picking k appropriately (i.e., a smaller k should be used for a problem with a smaller makespan), the k-splitting heuristic also yields near optimal makespans. As an example, for moving 100 robots on a 16×16 grid, BASEILP cannot produce a solution within 1000 seconds wheres the 4-splitting heuristic takes only 1.8 seconds on average. At the same time, the makespan obtained in this case by the 4-splitting heuristic is 25.5 steps. The reference line has a value of 25.2 - the difference is within two percent.

MPP_{p}. The k-splitting heuristic has some flexibility in selecting waypoints. This allows us to solve MPP_{p} via simulation by randomly trying different waypoints until we get a solution that contains no cyclic rotations. Figure 2 summarizes some of our results using this method (same problem instances used for MPP_{pr} were used here) on 8×8 and 16×16 grids.

Although this method for solving MPP_{p} is indirect, it still handily beats the current best minimum makespan MPP_{p} solver, iCOBOPT (Surynek 2012a; 2012b). As an example, iCOBOPT requires 910 seconds for computing an instance of MPP_{p} with 16 robots on a 16×16 grid. Using 2-splitting, our method can compute instances with 20 robots under one second, a roughly 1000x speedup. In this case, our method also produced a makespan of 22.6, which is optimal (since it is the same as the reference makespan). In general, our method has a speedup of about 100 times at the least and at the same time, produces plans with much better (smaller) makespan. Note that the computer hardware used from (Surynek 2012a; 2012b) is on par with ours.

References

Ding, J.; Chakrabarty, K.; and Fair, R. B. 2001. Scheduling of microfluidic operations for reconfigurable two-dimensional electrowetting arrays. *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems* 20(12):1463–1468.

Gurobi Optimization, I. 2012. Gurobi optimizer reference manual.

Kornhauser, D.; Miller, G.; and Spirakis, P. 1984. Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. In *Proceedings of the 25th Annual Symposium on Foundations of Computer Science (FOCS '84)*, 241–250.

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*, 23–28.

Standley, T., and Korf, R. 2011. Complete algorithms for cooperative pathfinding problems. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 668–673.

Surynek, P. 2012a. A sat-based approach to cooperative path-finding using all-different constraints. In *The 5th Annual Symposium on Combinatorial Search (SoCS)*.

Surynek, P. 2012b. Towards optimal cooperative path planning in hard setups through satisfiability solving. In *Proceedings 12th Pacific Rim International Conference on Artificial Intelligence*.

Wurman, P. R.; D'Andrea, R.; and Mountz, M. 2008. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI Magazine* 29(1):9–19.

Yu, J., and LaValle, S. M. 2013a. Planning optimal paths for multiple robots on graphs. to appear.

Yu, J., and LaValle, S. M. 2013b. Structure and intractability of optimal multi-robot path planning on graphs. to appear.