Finding Graph Topologies for Feasible Multirobot Motion Planning

Pushkar Kolhe Henrik I. Christensen

***Abstract*— In this paper we present a design methodology for mapping the configuration space of multirobot systems in a warehouse. The method generates a tree topology with robot and terminal nodes. In the most general case multirobot motion planning problems are NP-complete. But, we show that pebble motion problems in the resultant tree topology are always feasible. Our method uses an integer programming formulation to find such a solution.**

***Index Terms*— integer programming, graph theory, multi-robot motion planning**

I. Introduction

Roadmap methods reduce the workspace into a graph with nodes including the start and goals of the robot. It is possible to find a collision-free, probabilistic optimal path in this graph. Visibility graphs, vornoi diagrams and cell decomposition are other methods which reduce the workspace to a graph [1–3], but none of this methods are tailored for the multirobot motion planning (MRMP) scenarios. Space is a limiting constraint in a MRMP problem. Deadlocks pose a challenge when multiple robots have to execute crossing paths.

Our main goal is to find a minimal graph that is always *reachable*. Consider that the workspace can be reduced into a graph with nodes and edges. Each node can be occupied by a robot or it can be empty. Let there be n vertices and k robots located at distinct vertices in the graph. Each robot can be moved from its current position to an adjacent unoccupied vertex. Assume that each of the k robots has a new destination vertex to reach. Then the problem OPTIMAL(n, k) is called the *pebble motion problem* [4], which is to decide the least longest sequence of moves that transforms the robots from their original vertices to their destination vertices. For general graphs, the problem OPTIMAL(n, k) is known to be NP-complete [5]. Alternatively, one might ask if there is a sequence of moves that can lead the robots from their original to destination vertices? We denote this problem by FEASIBLE(n, k). We say that a graph is reachable if it is feasible for configurations of k robots in a graph with n vertices. There exist polynomial time algorithms to decide feasibility [6, 7].

We will consider the domain of multirobot fulfillment systems like Kiva [8]. The Kiva robots perform pick-and-fetch operations. They fetch the shelving unit from its storage location to a terminal, where a human handles packages kept in the unit. The units have to be placed in such a way that the pick and fetch operations are as efficient as possible. This includes ensuring that the space occupied by all units is small and the MRMP problem is always feasible. These two goals are often contradictory. Our objective is to generate a method that can design a topology in which the space used is minimal and at the same time all motion planning problems will always have a feasible solution.

The authors are with the Center for Robotics and Intelligent Machines & School of Interactive Computing at the Georgia Institute of Technology, Atlanta, GA 30332, USA. Emails: pushkar@cc.gatech.edu, hic@cc.gatech.edu

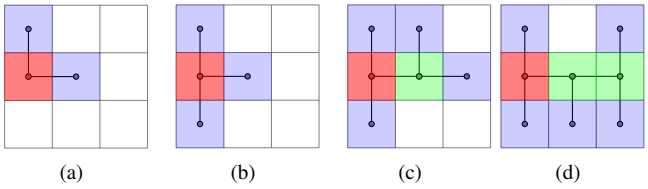


Fig. 1. Graph topology using our method for placing n = 2, 3, 4, 5 robots in a grid of size 3 and one terminal. The terminal nodes are marked with red, holes with green and possible robot locations with blue.

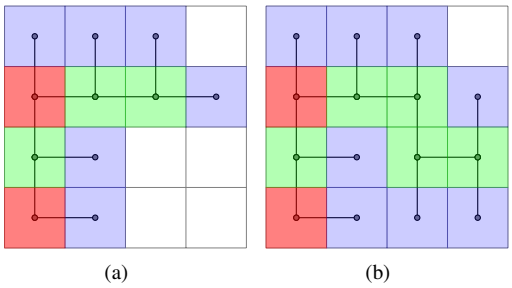


Fig. 2. Graph topology using our method for placing n = 6, 8 robots in a grid of size 4 and two terminals. The terminal is marked with red, holes with green and possible robot locations with blue.

II. Approach

The problem of finding an OPTIMAL(n, k) is NP-complete in general graphs [5] and grids [9]. However when the graph is a tree, OPTIMAL(n, k) can be found in $O(n^6)$ [10] or $O(n^5)$ [11]. In this paper we look at graph topologies which are trees. We use terminology and definitions from [12] to prove that the topology that our method determines is minimal.

We will use the example of multirobot fulfillment systems. Assume that the warehouse space can be divided into nodes. Each node can either be occupied, a hole or an empty space. The terminal nodes are holes which serve as destination nodes for a MRMP problem. Our main contributions are:

1) Find a tree topology which is always solvable for a multirobot motion planning problem.
2) Find a tree which is minimal in the physical space it occupies in the warehouse domain.
3) An integer programming (IP) formulation which can be easily extended for a large number of robots and a big warehouse.

We heavily rely on graph terminology from [13] throughout this paper.

III. Related Work

Kiva Systems [8] use a topology where the shelving units are stored in rectangular grids of a few robots. This configuration does not depend on the number of workstations or their position in the warehouse. In this configuration it is often hard to determine if a multirobot motion planning problem is feasible or not and deadlocks are common. With our approach to warehouse planning, deadlocks can be drastically reduced or completely eliminated.

Our solution is a graph topology for robots. Hence we compare our work with literature that deals with robot formation strategies. Most of these technique rely on dividing up the configuration space in a meaningful way. Sometimes it is necessary to sample the configuration space so that search problems are tractable. Composite roadmaps are used to decompose the configuration space into independent subcomponents which can be solved independently [14]. Heuristic based rapidly exploring random trees have been used to sample the configuration space for a navigation planning algorithm in multirobot systems [15], but it does not scale to a large number of robots as seen in the warehousing domain. Probabilistic roadmap techniques can be used to search in high-dimensional spaces [16], but they do not guarantee feasibility for multirobot motion planning.

To make the multirobot motion planning problems more tractable, graphs with specific topologies have been exploited. For example [17] takes advantage of the grid structure, [4, 18] introduces macros for efficient path planning on θ graphs and [19] discretizes space by decomposing the roadmap into subgraphs with specific properties. [20] makes path planning efficient by searching a minimum spanning tree of the roadmap. These approaches make an assumption about the graph structure, but our method is far more generic. It generates a generic tree topology that can be occupied within the configuration space and guarantees feasible motion plans.

An important issue in multirobot motion planning is to determine OPTIMAL(n, k) for any graph. The problem is addressed in works such as [6, 7, 21, 22]. We use [22] to analyze feasibility of our graph topologies.

IV. Problem Formulation

Consider a tree and let all arcs have assigned directions, so that we have a directed path from every node $i \neq j$ to node j. Such a directed graph will be called an *intree rooted at node j*. If it happens that for every $i \neq j$ along the tree is the shortest path, we say that we have a *tree of shortest paths*. We consider here the all-to-one shortest path problem.

We use a variation of the minimum spanning tree (MST) to formulate this problem. The MST is defined as a spanning tree such that the sum of the costs of its edges, for a given set of nodes is as small as possible. Our variation allows this formulation to choose a set of nodes that make the resultant graph feasible and is also minimal in size. We first introduce this formulation and then discuss optimality of motion planning for MRMP problems in this graph.

A. Minimum Spanning Trees (MST)

Let $G = (V, E)$ be an undirected graph with node set V ($|V| = n$) and edge set E. Every edge e has an associated cost c_e. The cost of the tree is the sum of the cost of the edges in the tree. The total number of edges in the tree should be $n - 1$. Moreover, the chosen edges should not contain a cycle. Let $S \subset V$. Then for the number of edges in S, we define

$$E(S) = \{\{i, j\} \in E \mid i \in S,\ j \in S\}.$$

Note that $E(V) = E$. This leads us to the *subtour elimination formulation* for the minimum spanning tree problem [13]:

$$\begin{aligned}
\min \quad & \sum_{e \in E} c_e x_e && \text{(1a)}\\
\text{s.t.} \quad & \sum_{e \in E} x_e = n - 1 && \text{(1b)}\\
& \sum_{e \in E(S)} x_e = |S| - 1, \qquad S \neq V, && \text{(1c)}\\
& x_e \in \{0, 1\}\ . && \text{(1d)}
\end{aligned}$$

Using an alternative, but equivalent definition, a tree is a connected graph containing $n - 1$ edges. We define a *cutset* $\delta(S)$ as

$$\delta(S) = \{\{i, j\} \in E \mid i \in S,\ j \notin S\}.$$

Here $\delta(\{i\})$ is the set of edges incident to i. We can express the connectivity requirement by replacing (1c) in terms of the constraints,

$$\sum_{e \in \delta(S)} x_e \geq 1, \quad S \neq V. \tag{2}$$

Both formulations have an exponential number of constraints. These problems can be solved in polynomial time by using plane cutting methods or interior point methods [13].

B. Variation of MST to find a topology

We now introduce binary variables y_i and z_i which represent if the node $\{i\} \in V$ is occupied or a hole. This new formulation allows the optimization problem to select nodes as holes to guarantee feasibility of a path from an occupied node to a terminal node. Consider

$$\min \quad \sum_{i \in V} c_i y_i + \sum_{i \in V} z_i + \sum_{e \in E} x_e \tag{3a}$$

$$\text{s.t.} \quad \sum_{e \in E} x_e = N(V) - 1 \tag{3b}$$

$$|S| \sum_{e \in E(S)} x_e = |S - 1|\ N(S), \quad |S| < |V|, \tag{3c}$$

$$\sum_{e \in \delta(S)} x_e \geq y_i + 2z_i, \qquad |S| = 1, \tag{3d}$$

$$\sum_{e \in \delta(S)} x_e \leq y_i + 4z_i, \qquad |S| = 1, \tag{3e}$$

$$\sum_{i \in V} y_i = n \tag{3f}$$

$$x_i + y_i <= 1 \tag{3g}$$

$$x_e \in \{0, 1\} \tag{3h}$$

$$y_i \in \{0, 1\} \tag{3i}$$

$$z_i \in \{0, 1\}\ , \tag{3j}$$

where $N(S)$ is the number of nodes which are either occupied or a hole. y_i or z_i have the value 1 only if y_i is occupied by a robot or z_i is a hole. The forcing constraint (3g) allows a node to be either be occupied or be a hole. Hence $N(S)$ gives the total number of nodes that should be a part of our graph. It can be defined by,

$$N(S) = \sum_{i \in S} y_i + \sum_{i \in S} z_i.$$

The subtour constraints (1c) is modified as (3c) to accommodate for constraints in which a node set is chosen which is empty. In the original MST, the subset can never be an empty nodeset. But in this formulation the chosen nodeset will remain empty if the objective function is minimal and all constraints are satisfied without requiring any node or edge from this subset. (3d) and (3e) are like the delta cut-set constraint for set of size 1. They put a lower and upper bound on the number of edges a robot node and a hole node can have. Without these constraints the graph topology might be infeasible. We prove this in the next section. The constraint (3f) ensures that at least n robot nodes are initialized for n robots. The problem is initialized by forcing the terminal workstations to be holes with $z_i = 1$.

C. Minimum Cost of this Graph

The objective function defined in (3a) defines the cost of the graph. Here we provide an intuition of this objective function based on minimum travel distance to all terminals.

***Definition** 1:* The cost of a node is the sum of the length of the path from its current position to all the terminals in the graph.

For the problem (3) to have a minimum the objective function used should follow the law of triangle inequality. To prove this we introduce the definition of steiner trees. A *steiner tree* is an undirected graph $G = (V, E)$ with nonnegative edge costs and whose vertices are partitioned into two sets, required and Steiner, and has a minimum cost tree that contains all the required vertices and a subset of the Steiner vertices. In our case the required vertices are the terminal nodes and Steiner vertices are the remaining nodes. All the nodes selected by the steiner tree will be on a path of minimum distance to reach any terminal vertex.

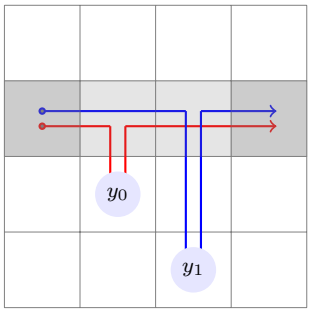


Fig. 3. The dark gray areas mark the terminals. The gray nodes from the Steiner tree. The cost of node y_0 and y_1 ($y_0 < y_1$) is shown in red and blue respectively. The figure shows that the cost function follows the law of triangle inequality.

As we can see from Figure 3 the distance y_1 has to travel more than y_0 is proportional to the Manhattan distance $\|y_1 - y_0\|$. Hence we can define the cost function as $d : V \to Q^+$ which gives the $L1$ norm from the vertex V to the nearest vertex of the steiner tree.

D. Feasibility of the Graph

We first provide some definitions from [12] before proving feasibility. A *Solvable Tree*, ST^n, is a tree on which any configuration of at most n robots can be reached from any initial configuration through their moves on tree edges.

The total number of vertices is called the *order* of the tree. The number of vertices connected to the vertex v is called its *degree*, $d(v)$. A *junction* is any vertex with degree $d(v) > 2$.

Let H be the number of holes in a graph. Then $H = |V| - n$, where $|V|$ is the order of the tree. Consider that a tree can be partitioned. We first show that a condition for which each these partitions is a solvable tree. Then we show conditions under which we can say that the combined set of all partitions is always solvable. For this we introduce these definitions from [12],

***Definition** 2:* Any two junctions are considered *Near* if there is no other Junction between them.

***Definition** 3:* The set of vertices within a distance of $H - 1$ around a Junction is called the *Influence Zone* (IZ) of that Junction.

If $Dist(.) : \{u, v\} \to Z^+$ is the Manhattan distance from vertex u to v, then IZ is defined as,

$$IZ_k = \{v | Dist(y_k, v) \leq H - 1,\ \forall\ v \in V\}. \tag{4}$$

***Definition** 4:* An *Interconnected Influence Zone* (IIZ) is the set of all IZs such that for any two Near IZs in the set, the distance between their Junctions is not more than (H-2).

$$IIZ_k = \{IZ_p | Dist(y_p, y_q) \leq H - 2\}. \tag{5}$$

Each IZ forms a partition of the tree. We state the following lemma which gives the condition for solvable trees.

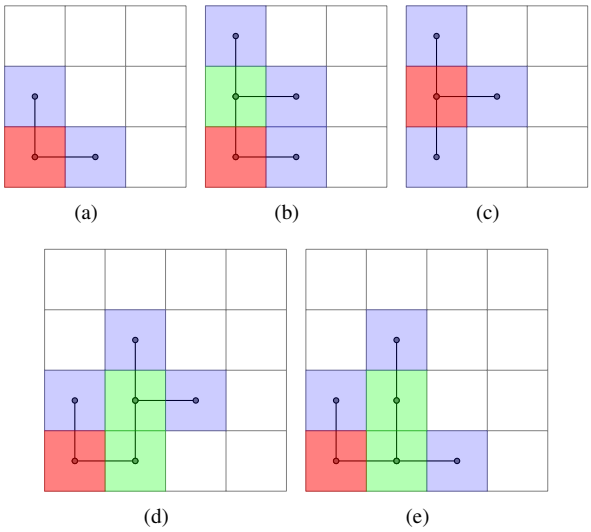


Fig. 4. Red are terminal nodes, blue are robot nodes and green are holes. (a) For $n = 2$ there is no junction node. (b, c) For $n = 3$ there is one junction node, but there can be 1 or 2 holes depending on the location of the terminal hole. (d) is not an optimal topology because (e) can be derived by moving a robot node and making the topology feasible. (e) has lesser cost than (d) because the Manhattan distance of the new node in (e) from the terminal node is less.

***Lemma** 1:* Any General Tree having H holes is solvable iff the following conditions are satisfied:

1) All two Near Influence Zones are Interconnected.
2) All vertices belong to at least one Influence zone.

Proof: A robot can move from any vertex to any other vertex within the same Influence Zone by creating enough holes from its starting node to goal node. We can show that a robot can create a temporary goal node within its current Influence Zone if it has to go into a new adjacent Influence Zone. Hence for a robot to always have a path to a goal node it should lie in an Influence Zone and all adjacent Influence Zones should be interconnected. A detailed proof of this lemma can be found in [12]. ■

***Theorem** 1:* A tree generated using (3) always admits Lemma 1.

Proof: We will use our definition of Influence Zone and the integer programming formulation and use induction to prove that these topological trees are in ST^n, where n is the number of robots. Consider the tree generated in Figure 4 for reference.

Consider the cutset constraint for every node in (3d) and (3e). The constraints puts a upper bound on the number of neighbors a robot node can have to one. The lower bound on the neighbors a hole can have is ≥ 2. This means that some holes in the tree are junctions, but no robot node is ever a junction.

1) Case with $n \leq 2$: Consider there is only one terminal node. There is no junction node as can be seen in Figure 4(a). According to (3b) the tree will at least have two edges. These two edges connect the robots to the terminal node directly to form a star graph. Every robot can reach the terminal node.

2) Case with $2 < n \leq 3$: If the terminal node is along the side of the configuration space then a star graph is formed. This is similar to the first case. If the terminal node is at the corner of the configuration space then there is at least one more hole other than the terminal node. See Figure 4(b) and Figure 4(c). The next case discusses this configuration.

3) Case with $n > 3$: In this case there is always a node which is a hole, but not a terminal. This means that $|H| \geq 2$. According to the definition of IZ, the hole h (which is a junction) has an influence zone which spans at least 1 edge.

Assume that h is connected to another hole (or the terminal node). If it is also a junction, then it has its own IZ. Then the graph can have 6 possible robot nodes around the two holes. It will admit to all conditions in Lemma 1. If one more robot is added to this configuration, then there will be at least one robot node with 2 neighbors which will violate the cutset constraint in (3e).

Assume that h is connected to another hole which is not a junction. Then h has at least 2 other neighbors and the new hole has only 1 other neighbor. In this configuration the tree is not solvable. Consider a new graph which we derive from this configuration by shifting a robot node from h to the new hole. The cost of this graph is always less than the original graph. See Figure 4(d) and Figure 4(e) Hence, either the new hole has to be a junction or it has to be connected to a third hole. If it is connected to a third hole, then the IZ of h is 2 edges. Again, the third hole can be a junction or not and the same argument follows.

From the above argument we can see that the new junction will always be within $|H| - 2$ edge distance. Hence all junctions are interconnected and all vertices are a part of the IZ of a junction node. Hence the tree admits Lemma 1 and is ST^n.

■

E. Solvability for any MRMP

A method to determine if any given general graph is solvable is shown in [22]. For this they introduce the concept of *Maximum Reachability Space* (MRS). According to their method if every robot has its destination node within MRS, the problem is feasible. In our case we can show that any two robot can exchange their positions in the graph. This means that the MRS of a robot is the entire graph.

Consider Figure 6. To exchange their locations, y_1 has to free up space for y_0. When y_0 reaches its destination, y_1 can go to its new destination. These exchanges are commutative in nature. Hence, if they cannot be exchange directly there will always be a robot node which both these nodes can exchange with.

Since for every robot node, the whole graph is MRS, any multirobot motion planning problem is solvable.

F. Experiments

We ran our integer programming formulation using the Gurobi Optimization library [23]. The library internally preprocesses the constraint matrix and reduces the size of the problem by using various plane cutting methods and interior point methods. Note that the number of constraints are exponential to the grid size. The size of the problem

is expressed in terms of the constraint matrix. The number of columns is the number of variables in the problem and the number of rows is the number of constraints used for optimization.

For a grid of size 3, Gurobi optimized a model with 479 rows and 30 columns to optimality in 0.033 seconds. We show these results in Figure 1. For a grid size of 4, Gurobi optimized a model with 64353 rows and 56 columns to optimality in 70 seconds on a dual core processor. We show these results in Figure 2.

V. Discussion

A. Motion Planning in this Graph

Since the graph is a *tree of shortest paths*, the shortest path from any node to a terminal can be found by a simple shortest path formulation. The topology of the graph is such that there are no obstacles along the shortest path from a robot node to any terminal node.

For any multirobot motion planning problem, we use an IP formulation of the network flow problem. We state the IP formulation for motion planning in this graph as a *multi-commodity flow problem* [13].

A multi-commodity flow problem can have many source nodes and many sink nodes. The flow problem models the flow through the network given the source and sink nodes alongwith the flow capacity constraints on the edges. Any multirobot motion planning problem can be converted to a *minimum-cost network flow program*. The advantage of solving multirobot problems with network flow is that the network simplex method can be solved faster than the general simplex method [13]. Such a formulation has:

1) Flow variables: They correspond to the flow in the edges of graph, x_e, $\forall e \in E$. If positive, the robot will go through the edge.

2) Flow conservation at the nodes: The total flow into a node equals the total flow out of a node. The robot can travel in either directions along an edge. We convert our undirected graph to a directed graph by introducing e new edges. When $b_i = 0$, the flow conservation constraint is given by (6).

$$b_i + \sum_{j \in I(e)} x_{ji} = \sum_{j \in O(e)} x_{ij} \tag{6}$$

3) Source and sink nodes: The source nodes correspond to the origin nodes of each robot. The sink nodes correspond to target nodes for each robot. For every source node $b_i = 1$ and for every sink node $b_i = -1$. If every edge cost is the true distance cost, then these values of b_i give us the shortest path solution to each individual robot plan. In our graph topology, the solution will be infeasible only if the number of sink nodes are less than the number of source nodes.

4) Bounds on arc flows: Each arc is bounded by the number of robots that can traverse it at the same time. In our case $0 \leq x_{ij} \leq 1$.

Data: x = edges, y, z = robot and hole nodes
Result: x
b = flow variables for each node;
foreach *node* i **do**
  $b_i = 0$;
  if i *is a Junction* **then**
    assign $b_i = -1$;
  end
  if i *is a Robot* **then**
    assign $b_i = 1$;
  end
end
x = network flow ip (x, y, z, b);

Fig. 5. Algorithm to navigate in a space efficient topology.

5) Objective function: The objective is to minimize the total cost of the flows over the network and reach a solution such that the robot are positioned close to the terminal nodes:

$$\min \quad \sum_{i \in V} c_i y_i + \sum_{e \in E} x_e \tag{7}$$

To solve the navigation problem the input is a set of source and sink nodes. The output of the network flow optimal solution is a set of x_e variables corresponding to each edge in the graph. If $x_{ij} = 1$, then the robot moves from node i to j.

VI. Conclusion

We have presented a technique to determine topological graphs for multirobot systems. We showed how we can use integer formulation techniques to find a feasible and optimal solution. Our method can also be scaled for a large number of robots. Instead of solving the NP-complete motion planning problem for multiple robots, we provide a novel approach. Instead of assuming any general graph, we show that we can design a graph topology over which preexisting network flow shortest path techniques can be used to generate feasible multirobot motion plans. Since our topology is feasible for any MRMP, we can use it to rearrange warehouse fulfillment systems. Rearragement brings frequently accessible robots near the terminal nodes.

VII. Future Work

A. Space efficient Graph Topology

The topology that we get from our original IP formulation of (3) is feasible and ST^n. But, it can be more efficient in occupying less space. From Theorem 1 we cannot reduce the number of vertices in our graph, but we can exchange the nearest holes with robot nodes. This gives more physical space and at the same time ensure that the robots can backup to their original position for making a MRMP feasible. We state an algorithm in Figure 5 to setup source and sink nodes and use the network formulation to get a space efficient topology. Figure 7 is an instance of 7 robots in a grid of size 4.

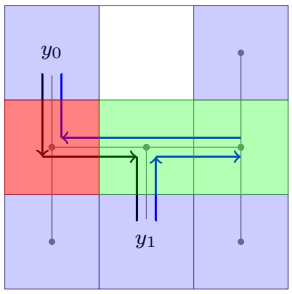
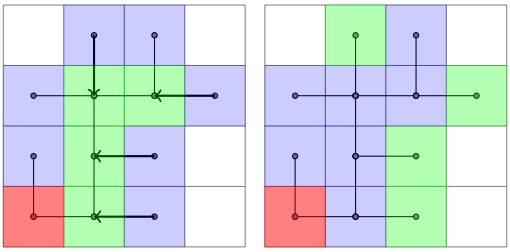


Fig. 6. A graph topology for a grid of size 3, 1 terminal (red) and 5 robots (blue). The graph shows a swap between y_0 and y_1. Such a swap can be implemented with any two robot nodes. Thus all MRMP problems are feasible.



(a) (b)

Fig. 7. Space efficient topology. All the farthest hole junctions can be filled by robot nodes until all are filled. This ensures that the physical space used by all the robots is minimum. Also, the shortest path to the terminal still remains the shortest. The obstacles are removed in the direction of their original locations.

B. Dynamic Graph Topologies

Our approach to this problem opens up new doors to solve stochastic optimization problems in this domain. We show that to make the multirobot network efficient it is necessary to redesign the topology of the graph. For example, by using the variation of the factory location problem of Karger and Minkoff [24], we can design a probabilistic model for the problem described by the IP formulation (3). Instead of using a greedy approach of storing every shelving unit near the center of the warehouse, as we do now, it is more efficient to design a graph that spreads into the warehouse and provides easy access points near the terminal workstations. Pareto optimality objectives likes these are surveyed in [25]. It might be more efficient to set up terminals so that they serve only a few inventory items. This raises many interesting questions on how this method can be modified and improvised upon.

C. Approximate Optimization for IP

In the IP formulation that we suggested, the number of constraints increase with the size of the problem. To make the problem more feasible we can use plane-cutting methods to presolve and remove constraints which do not affect the optimal value. Then we can use interior point methods to find an optimal solution.

In a similar way, we can use parallel integer programming and other approaches from the Integer Programming literature to make running time for a large problem more feasible.

References

[1] S.M. LaValle. *Planning algorithms*. Cambridge Univ Pr, 2006. 1

[2] Jean-Claude Latombe. *Robot motion planning*. Springer Verlag, 1990. 1

[3] H.M. Choset. *Principles of robot motion: theory, algorithms, and implementation*. The MIT Press, 2005. 1

[4] P. Surynek. An application of pebble motion on graphs to abstract multi-robot path planning. In *Tools with Artificial Intelligence, 2009. ICTAI'09. 21st International Conference on*, pages 151–158. IEEE, 2009. 1, 2

[5] O. Goldreich. Shortest move-sequence in the generalized 15-puzzle is np-hard. *Manuscript, Laboratory for Computer Science, MIT*, 1984. 1

[6] V. Auletta, A. Monti, M. Parente, and P. Persiano. A linear-time algorithm for the feasibility of pebble motion on trees. *Algorithmica*, 23(3):223–245, 1999. 1, 2

[7] D.M. Kornhauser, G. Miller, and P. Spirakis. Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. Master's thesis, M. I. T., Dept. of Electrical Engineering and Computer Science, 1984. 1, 2

[8] J.J. Enright and P.R. Wurman. Optimization and coordinated autonomy in mobile fulfillment systems. In *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011. 1, 2

[9] D. Ratner and M. Warmuth. Finding a shortest solution for the n× n-extension of the 15-puzzle is intractable. *J. Symb. Comp*, 10:111–137, 1990. 1

[10] C.H. Papadimitriou, P. Raghavan, M. Sudan, and H. Tamaki. Motion planning on a graph. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, pages 511–520. IEEE, 1994. 1

[11] V. Auletta and P. Persiano. Optimal pebble motion on a tree. *Information and Computation*, 165(1):42–68, 2001. 1

[12] E. Masehian and A.H. Nejad. Solvability of multi robot motion planning problems on trees. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 5936–5941. IEEE, 2009. 1, 3, 4

[13] D. Bertsimas and J. Tsitsiklis. introduction to linear optimization. *IIE Transactions*, 30:855–863, 1998. 2, 5

[14] J. van den Berg, J. Snoeyink, M. Lin, and D. Manocha. Centralized path planning for multiple robots: Optimal decoupling into sequential plans. *Robotics: Science and Systems V*, 2009. 2

[15] J. Bruce and M. Veloso. Real-time randomized path planning for robot navigation. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 3, pages 2383–2388. Ieee, 2002. 2

[16] L.E. Kavraki, P. Svestka, J.C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE Transactions on*, 12(4):566–580, 1996. 2

[17] K.H.C. Wang and A. Botea. Tractable multi-agent path planning on grid maps. In *Proceedings of the International Joint Conference on Artificial Intelligence IJCAI-09*, pages 1870–1875, 2009. 2

[18] N. Sturtevant and M. Buro. Improving collaborative pathfinding using map abstraction. In *The Second Artificial Intelligence for Interactive Digital Entertainment Conference (AIIDE06)*, pages 80–85, 2006. 2

[19] MRK Ryan. Graph decomposition for efficient multi-robot path planning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2003–2008, 2007. 2

[20] M. Peasgood, C.M. Clark, and J. McPhee. A complete and scalable strategy for coordinating multiple robots within roadmaps. *Robotics, IEEE Transactions on*, 24(2):283–292, 2008. 2

[21] M.R.K. Ryan. Exploiting subgraph structure in multi-robot path planning. *JAIR*, 31(1):497–542, 2008. 2

[22] E. Masehian, H. Samadian, and F. Daneshzand. Analysis of motion feasibility of multiple mobile agents on graphs. In *Proc. 4th International Conference on Information Technology (ICIT)*, 2009. 2, 4

[23] R. Bixby, Z. Gu, and E. Rothberg. Gurobi optimization, 2010. 4

[24] DR Karget and M. Minkoff. Building steiner trees with incomplete global knowledge. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 613–623. IEEE, 2000. 6

[25] R. Ghrist, J.M. OKane, and S.M. LaValle. Pareto optimal coordination on roadmaps. *Algorithmic Foundations of Robotics VI*, pages 171–186, 2005. 6