2011 IEEE/RSJ International Conference on
Intelligent Robots and Systems
September 25-30, 2011. San Francisco, CA, USA

M*: A Complete Multirobot Path Planning Algorithm with Performance Bounds

Glenn Wagner, Howie Choset

***Abstract*— Multirobot path planning is difficult because the full configuration space of the system grows exponentially with the number of robots. Planning in the joint configuration space of a set of robots is only necessary if they are strongly coupled, which is often not true if the robots are well separated in the workspace. Therefore, we initially plan for each robot separately, and only couple sets of robots after they have been found to interact, thus minimizing the dimensionality of the search space. We present a general strategy called subdimensional expansion, which dynamically generates low dimensional search spaces embedded in the full configuration space. We also present an implementation of subdimensional expansion for robot configuration spaces that can be represented as a graph, called M*, and show that M* is complete and finds minimal cost paths.**

I. Introduction

Multirobot systems are attractive for surveillance, search and rescue, and warehouse automation applications. Unfortunately, the flexibility and redundancy that make multirobot systems attractive also make planning for such systems difficult. Handling a high dimensional configuration space is the fundamental problem of multirobot path planning.

Multirobot path planning algorithms can be divided into two categories: coupled and decoupled [15]. A coupled algorithm seeks to find a path in the full configuration space of a system [1][3][7]. While the full configuration space contains all possible paths, it grows exponentially with the number of robots in the system. As a result, coupled planners may be guaranteed to find an optimal path, but are computationally infeasible for systems of many robots.

On the other hand, decoupled algorithms search one or more low dimensional search spaces which represent a portion of the full configuration space [6][9][16][18][21]. Searching a lower dimensional representation reduces the computational cost of finding a path, but the representation may not capture some or all of the solutions to the planning problem. As a result, decoupled algorithms generally produce results more quickly, but the quality or existence of the solution is not guaranteed.

This paper presents an approach that shares the benefits of both coupled and decoupled approaches, which we term *subdimensional expansion*. Subdimensional expansion initially uses decoupled planning to generate a low-dimensional search space. As robot-robot collision are found in the search space, the local dimensionality of the space is locally increased to construct a search space of minimal size that contains a path with the desired properties.

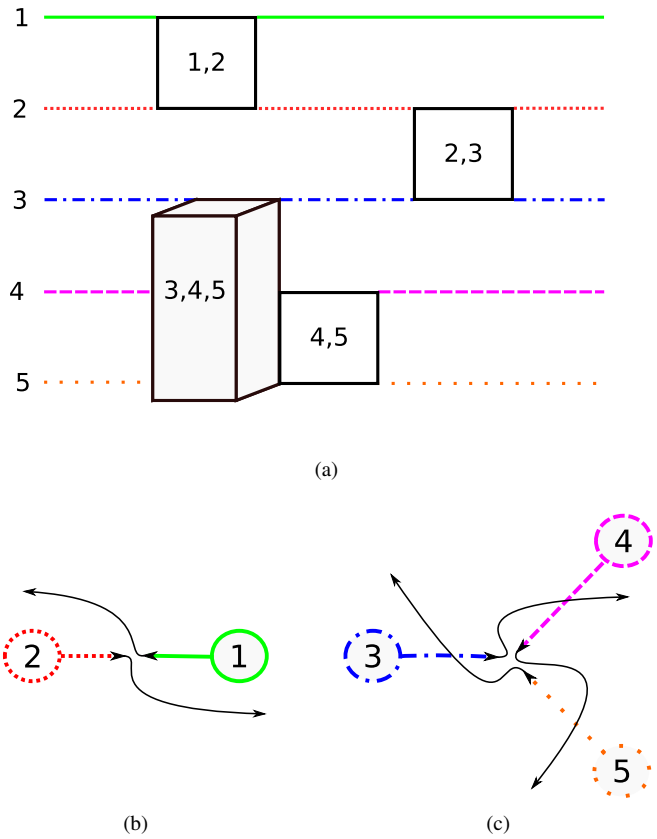

Fig. 1: A conceptual visualization of a variable dimensionality search space for five robots (a). Initially each robot is constrained to its individually optimal path, represented by a single line, but when robots 1 and 2 collide (b), the local dimensionality of the search space must be increased, as represented by a square. When three robots collide while following their individually optimal paths (c), the local dimensionality of the search space must be increased further, represented by the cube, to include all local paths of the three robots.

We implement subdimensional expansion for configuration spaces which can be represented as graphs, using A* as the underlying path planning algorithm. We name the resulting algorithm *M**. We prove that M* is complete and optimal, then show in simulation that M* requires dramatically less time than A* to find paths for multirobot systems.

II. Prior Work

A number of algorithms exist that dynamically vary how robots are coupled for planning purposes. Krishna *et al.* developed an approach for decentralized dynamic coupling

Glenn Wagner is a graduate student at the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 15213 gswagner@andrew.cmu.edu
Howie Choset is an associate professor at the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 15213 choset@cs.cmu.edu

of robots for velocity planning [13]. In their algorithm, robots first try to resolve a potential collision by independently altering their velocity. If this does not succeed, the robots involved in the collision cooperate to find a safe velocity schedule. If this also fails, they recruit uninvolved robots to alter their velocities to allow for a solution to be found. This approach will never change the spatial path the robots follow, and thus is neither complete nor optimal.

Clark *et al.* introduced dynamic networks, which explicitly search configuration spaces of varying sizes[5]. Joint plans are computed for groups of robots capable of mutual communication. Paths are replanned whenever a new robot joins the group. This approach will lead to unnecessary coupling, as not all robots that can communicate need to cooperate to find a safe path, and only considers local interactions.

Van den Berg *et al.* [23] developed a planning time algorithm to find a coupling strategy that minimizes the size of the largest set of coupled robots needed to guarantee that a solution will be found. The robots are constrained to move sequentially, which induces constraints on which robots must move before or after other robots. Cycles in these constraints can be used to find sets of robots for which coupled planning is necessary. This approach is non-ideal due to the restrictions it places on robot motion, which results in non-optimal paths, and the global nature of the coupling it performs.

There has also been work in the machine learning community to determine when coupling multiple robots is necessary. Kok *et al.* [11] presented an approach which performs Q-learning for robots individually, but stores statistics for the reward of the joint actions that are explored. If these statistics indicate that coordinating actions at a specific space is beneficial, then the algorithm starts learning coordinated actions at that state. This approach has the benefit of being able to handle tasks besides basic path planning, such as capturing targets that required coordinated action by multiple pursers. Melo and Veloso [19] developed a Q-learning algorithm that adds a 'coordinate' action to the set of actions available to each robot, which uses the state of the nearest neighboring robot to help choose the action to perform. Coordination between robots then only occurs when a robot learns to take the coordinate action. Our work focuses on dynamic coupling in the context of search, rather than reinforcement learning.

III. Subdimensional Expansion

A. Problem Statement

We seek to find an optimal, collision free path for a set of n robots in a common workspace $\mathcal{W}$, from a specified initial configuration to a goal configuration. We index the robots r^i with the set $I = \{1, \ldots, n\}$. For brevity, we conflate robots with their indices. We use superscripts to indicate which robots are described by a set, space, or element, while subscripts indicate a specific position in a space or a specific element of a set.

Each robot r^i has an obstacle free configuration space Q^i. The full system has an obstacle free configuration space $Q = \prod_{i \in I} Q^i$, although states that result in robot-robot collisions remain in Q. We will deal with subspaces frequently, so we use the notation $Q^\Omega = \prod_{i \in \Omega} Q^i$, to denote the joint configuration space of the subset of robots $\Omega \subset I$. The same notation is used to describe paths of subsets of robots, the location of a subset of robots, and so forth.

Since each point $q_k \in Q$ simultaneously describes the position of every robot, a path in Q implicitly provides temporal coordination. Time can be added as an element of the robot state if the cost function depends on time. Doing so results in a minimal performance penalty, since all robots have the same time dynamics. Therefore, the time dimensions of each robot will collapse into a single effective dimension when we search Q.

We use $\pi(q_k, q_l)$ to represent the set of points along a path from $q_k \in Q$ to $q_l \in Q$, and $\pi^*(q_k, q_l)$ to denote a collision free path that minimizes a specified cost function. Our goal is to find $\pi^*(q_I, q_F)$, a optimal collision free path from the initial configuration q_I to the goal configuration q_F.

We wish to minimize the cost function $f(\pi(q_k, q_l))$. We assume that the cost of the path in the full configuration space is the sum of the costs of the paths of the individual robots[1]

$$f(\pi(.)) = \sum_{i \in I} f^i(\pi^i(.)) \qquad \pi^i(.) \subset Q^i \tag{1}$$

The form of (1) ensures that the cost function can be decomposed into a separate cost function for each robot. Furthermore, no path $\pi(q_I, q_F) \subset Q$ can be cheaper than the path formed by separately optimizing the path of each individual robot. While such a path almost certainly contains at least one robot-robot collision, it is still useful for guiding the search for a collision free path.

To ensure that any path of finite cost has finite length, we further require that

$$\exists\, \epsilon > 0 \; s.t. \; f^i(\pi^i(q_k^i, q_l^i)) > \epsilon \qquad q_k^i, q_l^i \in Q^i \tag{2}$$

for all paths which do not always remain at q_F^i.

We define a collision function Ψ^{ij} for $i \neq j \in I$.

$$\Psi^{ij}(q^i, q^j) = \begin{cases} \{i, j\}, & A(q^i) \cap A(q^j) \neq \emptyset \\ \emptyset, & \text{otherwise} \end{cases} \tag{3}$$

where $A(q^i)$ is the subset of $\mathcal{W}$ occupied by r^i when located at $q^i \in Q^i$. We define a global collision function $\Psi : Q \to I$, which is the union of all pairwise collision functions.

$$\Psi(q) = \bigcup_{i \neq j \in I} \Psi^{ij}(q^i, q^j) \tag{4}$$

The form of (4) means for q_k and q_k' formed from q_k by changing the coordinates of one or more robots not in $\Psi(q_k)$, any robot that is in collision with another robot at q_k will remain in collision at q_k', i.e. $\Psi(q_k) \subset \Psi(q_k')$. As a result, if the robots r^i and r^j collide along some path $\pi(.)$, the paths of r^i or r^j must be altered to produce a collision free path. We "overload" the collision function to apply to paths, $\Psi(\pi(.)) = \bigcup_{q \in \pi(.)} \Psi(q)$. The constraint that $\pi^*(q_I, q_F)$ must be collision free can be expressed as $\Psi(\pi^*(q_I, q_F)) = \emptyset$.

[1] We further assume that the cost of a path from q_k does not depend on the path taken to q_k. If this would be violated, we can add path history to the state of the robots.

B. Approach

Subdimensional expansion exploits the natural decoupling of robots in systems which satisfy (1) and (4) to construct a sufficient low dimensional search space $Q^{\#}$ embedded in Q. We use a path planning algorithm, referred to as the planner, to search $Q^{\#}$. As the planner searches $Q^{\#}$, it will find information about robot-robot collisions, which is then used to locally augment the dimensionality of $Q^{\#}$. In this manner, we tailor the search space to the structure of the problem at hand, allowing us to search a low dimensional space, while guaranteeing that the desired path will eventually exist within the search space.

For each robot, we define an individually optimal policy $\phi^i : Q^i \to TQ^i$ which maps the position of a robot to its motion. We choose ϕ^i such that the path induced by obeying ϕ^i from any point $q_k^i \in Q^i$ is an optimal path to $q_F^i \in Q^i$. We denote such a path $\pi^{\phi^i}(q_k^i, q_F^i)$.

We use the notation $\phi^{\Omega}(q^{\Omega}) = \prod_{i\in\Omega} \phi^i(q^i)$ to denote the individually optimal policy for a subset of robots Ω, and use $\phi(q) = \prod_{i\in I} \phi^i(q^i)$ when $\Omega = I$. We use $\pi^{\phi^{\Omega}}(q^{\Omega}, q_F^{\Omega})$ and $\pi^{\phi}(q, q_F)$ to denote the paths induced by ϕ^{Ω} and ϕ respectively. We term such paths *individually optimal paths*. We note that by the form of (1), $f(\pi^{\phi}(q_k, q_F))$ is a lower bound on the costs of all paths $\pi(q_k, q_F)$.

At each instant during the search, we take the optimistic view that the individually optimal path from q_k will be collision free, unless we have specific information to the contrary. We maintain a *collision set* C_k for each $q_k \in Q$, which is the set of robots for which the optimistic view at q_k has been invalidated. Let $\Pi(q_k)$ be the set of paths the planner has searched that pass through q_k. Then C_k is defined as

$$C_k = \bigcup_{\pi\in\Pi(q_k)} \Psi(\pi) \tag{5}$$

The collision set C_k thus consists of all robots r^i for which the planner has found a path from q_k to a collision containing r^i (Figure 2).

We wish to restrict the set of robots for which we maintain the optimistic view, $\bar{C}_k = I \backslash C_k$, to their individually optimal path, in line with our optimistic belief that this path is collision free. However, we will place no such restriction on the robots in C_k, as we already know that the optimistic view point does not hold. We encode these constraints in the geometry of the search space $Q^{\#}$ by proper choice of the tangent space $T_{q_k}Q^{\#}$ of $Q^{\#}$ at q_k.

$$T_{q_k}Q^{\#} = \mathbf{t}^{\bar{C}}(q_k) \times \prod_{j\in C_k} T_{q_k^j}Q^j \tag{6}$$

We restrict the subset of robots $\bar{C}_k$ to move in the direction of the vector $\mathbf{t}^{\bar{C}}(q_k)$ which is tangent to the individually optimal path for that subset of robots, $\pi^{\phi^{\bar{C}_k}}(q_k^{\bar{C}_k}, q_F^{\bar{C}_k})$, at q_k. This locally restricts the robots in $\bar{C}_k$ to their individually optimal paths. We need to perform exhaustive search for all robots in C_k, and thus must consider the set of all directions in which such r^i can move, $T_{q_k^i}Q^i$.

We can now construct $Q^{\#}$ by starting at q_I and using the definition of the tangent space to differentially grow $Q^{\#}$. Initially, $Q^{\#}$ will be $\pi^{\phi}(q_I, q_F)$, and will then grow along various subspaces as the planner discovers robot-robot collisions.

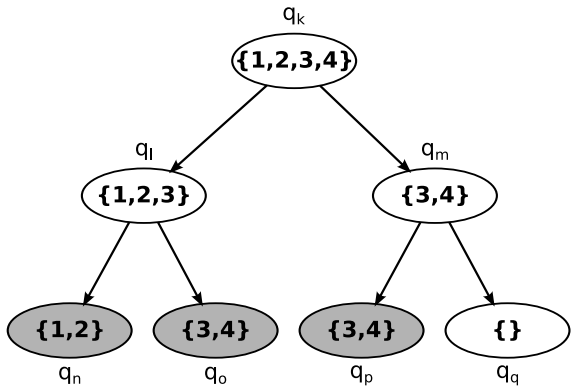

Fig. 2: Representation of a search tree and resultant collision sets. Ovals represent configurations in Q. Arrows represent searched path from the higher configuration to the lower configuration. If there is a robot-robot collision at a state, the oval is gray. The set contained inside the oval represents the collision set. Since there is a searched path from q_k to q_n, q_o, and q_p, C_k contains all robots which collide at the aforementioned state. Since subdimensional expansion has not found a path from q_q to any state with a collision, C_q is empty

IV. M*

A. Description

M* is an implementation of subdimensional expansion for cases where the configuration space of each robot r^i can be represented by a directed graph $G^i = \{V^i, E^i\}$. V^i is the set of vertices in G^i that represent positions in Q^i, while E^i is the set of directed edges e_{kl}^i which represent valid transitions connecting $v_k^i \in V^i$ to $v_l^i \in V^i$. We make no assumption about the representation used, so G^i may be an approximate cellular decomposition, a generalized Voronoi diagram, or other graph representation of the configuration space. We represent the full configuration space of the system with the graph $G = \{V, E\} = \prod_{i\in I} G^i$. The Cartesian product of two graphs, $G^i \times G^j$, has the vertex set $V^i \times V^j$, and the edge e_{kl} is in the edge set if $e_{kl}^i \in E^i$ and $e_{kl}^j \in E^j$. The vertex in G which represents the initial configuration of the system is denoted v_I, while the goal configuration is denoted v_F.

Representing the configuration space as a graph converts the path planning problem into a graph search problem. This allows us to base M* on A*, a complete and optimal graph search algorithm [8]. Recall that A* maintains an *open list* of vertices v_k to explore. These are sorted based on the sum of the cost of the cheapest path $\pi(v_I, v_k)$ and a *heuristic cost*, which is a lower bound on the cost of any path $\pi(v_k, v_F)$. At each iteration, the most promising vertex, v_k, is taken from the open list and expanded. For each neighbor v_l of v_k, A* checks whether reaching v_l via v_k is the cheapest path found thus far to v_l. If so, v_l is added to the open list. This continues until v_F is expanded, indicating that an optimal path to the goal has been found.

Algorithm 1 backprop(v_k, C_l,open):
v_k- vertex in the backpropagation set of v_l
C_l- the collision set of v_l
open- the open list for M*

if $C_l \not\subset C_k$ **then**
  $C_k \leftarrow C_k \bigcup C_l$
  if $\neg(v_k \in$ open) **then**
    open.append(v_k) {If the collision set changed, we will need to re-expand v_k}
  for $v_m \in v_k$.back_set **do**
    {Iterate over the backpropagation set}
    backprop(v_m, C_k,open)

M* is similar to A*. However, in the expansion step, M* only considers the *limited neighbors* of v_k, a subset of the neighbors of v_k in G, determined by C_k. The set of limited neighbors $\hat{V}_k$ is the set of vertices v_l which can be reached from v_k while moving each robot $r^i \in \bar{C}_k$ according to its individually optimal policy $\phi^i(v_k^i)$, where v_k^i is the position of r^i when the system is at v_k. Conversely, the robots $r^j \in C_k$ are allowed to move to *any* neighbor of v_k^j in Q^j

$$\hat{V}_k = \left\{ v_l | \forall i \in I,\ v_l^i\ s.t. \left\{ \begin{array}{ll} e_{kl}^i \in E^i, & i \in C_k \\ v_l^i = \phi^i(v_k^i), & i \notin C_k \end{array} \right\} \right. \quad (7)$$

If $\Psi(v_k) \neq \emptyset$, we set $\hat{V}_k = \emptyset$, to prevent M* from considering paths which pass through collisions.

We need an efficient method for keeping the collision sets updated, which is achieved by passing information about a collision back along all searched paths that reach the collision. To do this, we maintain a *backpropagation set* for each vertex v_k, which is the set of all vertices v_l which were expanded while v_k was in $\hat{V}_l$. The backpropagation set is thus the set of neighbors of v_k through which the planner has found a path to v_k We propagate information about a collision at v_k by adding $C_k = \Psi(v_k)$ to C_l for each v_l in the backpropagation set of v_k. We then add C_l to the collision set of each vertex in the backpropagation set of v_l, and repeat this process until a collision set is encountered which contains C_k. Since $\hat{V}_l$ is dependent on C_l, changing C_l adds new paths through v_l to the search space. As a result, v_l must to be added back to the open list so that these new paths can be searched (See algorithm 1).

Finally, we note that since $f(\pi^\phi(v_k, v_F))$ is a lower bound on the cost of all paths $\pi(v_k, v_F)$, it is an obvious choice for use as the heuristic function for M*. We denote the heuristic function

$$h(v_k) = f(\pi^\phi(v_k, v_F)) \leq f(\pi^*(v_k, v_F)) \quad (8)$$

M* is described in algorithm 2.

B. Graph-Centric Description

The description of M* given in IV-A provides a local description of the search process. We now present an alternate description which better captures the global properties of M*, but is not suited to implementation.

Algorithm 2 Pseudocode for M*

for all $v_k \in V$ **do**
  v_k.cost $\leftarrow$ MAXCOST
  $C_k \leftarrow \emptyset$
v_I.cost $\leftarrow 0$
v_I.back_ptr $= \emptyset$
open = $\{v_I\}$
while True **do**
  open.sort() {Sort in ascending order by v.cost + h(v)}
  v_k = open.pop(0)
  if $v_k = v_I$ **then**
    {We have found a solution}
    return back_track(v_k) {Reconstruct the optimal path by following v_k.back_ptr}
  if $\Psi(v_k) \neq \emptyset$ **then**
    CONTINUE {Skip vertices in collision}
  for $v_l \in \hat{V}_k$ **do**
    v_l.back_set.append(v_k) {Add v_k to the back propagation list}
    $C_l \leftarrow C_l \bigcup \Psi(v_l)$
    {Update collision sets, and add vertices whose collision set changed back to open}
    backprop(v_k, C_l,open)
    if v_k.cost+$f(e_{kl}) < v_l$.cost **then**
      {We have found a cheaper path to v_l}
      v_l.cost $\leftarrow$ v_k.cost+$f(e_{kl})$
      v_l.back_ptr $\leftarrow$ v_k {Keep track of the best way to get here}
return No path exists

When examining algorithm 2, we see that M* differs from A* only in the existence of the **backprop** function, and the use of $\hat{V}_k$ in the place of all neighbors of v_k in G when exploring paths from v_k. The backprop function only has a non-trivial result when a new path to one or more collisions is found. Therefore, M* behaves exactly like A* running on a graph $G^\#$ where the neighbors of v_k in $G^\#$ are the vertices in $\hat{V}_k$, until A* finds a new robot-robot collision. By thinking of M* as alternating between running A* on $G^\#$ and updating $G^\#$ based on the search results, we can exploit the optimality and completeness properties of A* to prove similar properties of M*.

$G^\#$ consists of three subgraphs: G', $\hat{G}$, and G^ϕ. G' is the portion of $G^\#$ which has been searched by M*, $\hat{G}$ represents the limited neighbors of the vertices in G', and G^ϕ connects the vertices in $\hat{G}$ to v_F by obeying ϕ.

The portion of G which has been searched by M* is represented by the graph $G' = \{V', E'\}$. V' is the set of vertices which have been added to the open list. E' consists of the directed edges e_{kl} connecting each vertex v_k which has been expanded by M* to the vertices $v_l \in \hat{V}_k$. Since G' represents all paths which have been searched by the planner, we can use G' to define the collision set

$$C_k = \left\{ \begin{array}{ll} \Psi(v_k) \bigcup_{v_l\ s.t.\ \exists \pi(v_k, v_l) \subset G'} \Psi(v_l) & v_k \in G' \\ \emptyset & v_k \notin G' \end{array} \right. \quad (9)$$

If $v_k \notin G'$, then M* has never visited v_k, and we have never

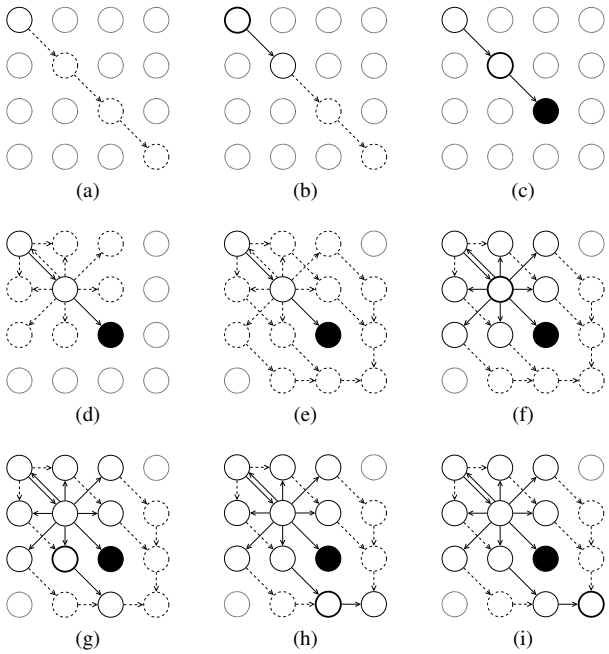

Fig. 3: The above figure shows how G' and $G^{\#}$ evolve in the configuration space of two, one-dimensional robots. Vertices are represented as circles, with arrows representing directed edges. G' is denoted by solid lines, while $G^{\#} \backslash G'$ is shown as dashed lines. $G \backslash G^{\#}$ is represented by dotted lines, with edges suppressed for clarity. A vertex is given a bold outline when it is expanded, while filled circles represent vertices with known robot-robot collisions. v_I is in the upper left, while v_F is in the bottom right. In (a), (b) and (c), the most promising vertex in the open list is expanded, until a collision is found. $\hat{G}$ is updated to reflect the new collision sets in (d). G^{ϕ} is then updated in (e). In (f) a vertex is re-expanded, having been added back to the open list when its collision set was changed. (g), (h) and (i) see the most promising vertices in the open list expanded, until v_F is expanded, indicating that a path has been found.

computed $\Psi(v_k)$. Until M* actually visits v_k, we take the optimistic view that v_k and $\pi^{\phi}(v_k, v_F)$ are collision free, and thus set $C_k = \emptyset$ and $\hat{V}_k = \phi(v_k)$.

We represent the portion of the graph which will be explored when v_k is expanded by the graph $\hat{G}_k$, which is the graph formed from v_k, its limited neighbors $\hat{V}_k$, and the edges connecting v_k to the vertices in $\hat{V}_k$. Let $\hat{G} = \bigcup_{v_k \in G'} \hat{G}_k$.

Since $C_k = \emptyset$ for all v_k which are not in G',, we know that the search from v_k will be constrained to $\pi^{\phi}(v_k, v_F)$ as long as this path lies entirely outside of G'. Let the graph G_k^{ϕ} represent the portion of $\pi^{\phi}(v_k, v_F)$ from v_k to the first vertex along the path in G', or v_F if $\pi^{\phi}(v_k, v_F)$ never reenters G'.

We can now define $G^{\#}$ as

$$G^{\#} = G' \bigcup_{v_k \in G'} \left(\hat{G}_k \bigcup_{v_l \in \hat{G}_k \backslash G'} G_l^{\phi} \right) \tag{10}$$

As a result of our definitions of G', $\hat{G}$ and G^{ϕ}, vertices and edges shift from G^{ϕ} to $\hat{G}$, and from $\hat{G}$ to G' as M* searches $G^{\#}$. See Figure 3 for an illustration of how the subgraphs change over time. However, $G^{\#}$ as a whole only changes when the collision set of a vertex in $G^{\#}$ changes.

V. Completeness and Cost-Optimality

A path planning algorithm is *complete* if it is guaranteed to either find a path or to determine that no path exists, in finite time [4]. We also wish to prove that M* is cost optimal, meaning that M* will find a path that minimizes a cost function. As demonstrated in IV-B, we can treat M* as alternating between running A* search on a graph $G^{\#}$ and modifying $G^{\#}$ based on the results of the A* search. Since A* is complete and cost optimal, we can prove that M* is complete and cost optimal if we demonstrate that $G^{\#}$ will contain $\pi^*(v_I, v_F)$ after a finite number of modifications or, if $\pi^*(v_I, v_F)$ does not exist, $G^{\#}$ will be modified at most a finite number of times.

We first assume that no solution exists, and show that M* will terminate in finite time without finding a path. $G^{\#}$ is only modified when the collision set of at least one vertex in $G^{\#}$ is modified. One or more robots are added to a collision set whenever it is modified, so each collision set can be modified at most $n-1$ times, since the fist modification must add at least two robots. Therefore, $G^{\#}$ can be modified at most $(n-1) * |G|$ times. We know that A* will expand each vertex of a given graph at most once [8]. Therefore, M* will process each iteration of $G^{\#}$ in finite time, so M* will terminate in finite time.

$G^{\#}$ may contain a path $\pi(.)$ that has a robot-robot collision at v_k. This can only occur if $v_k \notin G'$, as otherwise a vertex which contains a robot-robot collision will not have any out-neighbors. However, before M* can return such a path, v_k will be added to the open list, and thus to G'. As a result, $G^{\#}$ will be modified to remove the out-neighbors of v_k, thus removing the invalid path. Therefore, M* will never return a path containing a collision. We can conclude that if no solution exists, M* will determine that no valid path exists in finite time.

Next, assume that a path from v_I to v_F exists. We will first show that if one of two cases is always true, M* will find an optimal, collision free path. We will then show that one of these two cases must always hold. Assume that $G^{\#}$ always contains either

Case 1: an optimal, collision free path, $\pi^*(v_I, v_F)$
Case 2: a path $\pi(v_I, v_k)$ s.t. $f(\pi(v_I, v_k)) + h(v_k) \leq f(\pi^*(v_I, v_F))$, and $\exists\ v_l \in \pi(v_I, v_k)$ s.t. $\Psi(v_k) \not\subset C_l$

If case 1 holds, running A* on $G^{\#}$ will find $\pi^*(v_I, v_F)$, unless there exists a cheaper path $\tilde{\pi}(v_I, v_F) \subset G^{\#}$. If $f(\tilde{\pi}(v_I, v_F)) < f(\pi^*(v_I, v_F))$ then by the definition of $\pi^*(.)$, there must be a vertex $v_k \in \tilde{\pi}(v_I, v_F)$ such that $\Psi(v_k) \neq \emptyset$. By Eq (8), $f(\tilde{\pi}(v_I, v_k)) + h(v_k) < f(\tilde{\pi}(v_I, v_F)) < f(\pi^*(v_I, v_F))$. v_k must be in $G^{\#} \backslash G'$, as otherwise v_k would have no out-neighbors, and thus no path could pass through v_k. Since v_k is not in G', $C_k = \emptyset$, and therefore $\Psi(v_k) \not\subset C_k$. As a result, v_k fulfills the role of both v_l and v_k in case 2, so case 2 is satisfied. We can conclude that if case 1 holds, then M* will find $\pi^*(v_I, v_F)$ unless case 2 also holds.

If case 2 holds, then v_k will be added to G' before A* finds any path to v_F that costs more than $f(\pi^*(v_I, v_F))$ [8].

Adding v_k to G' will cause C_l to be modified, changing $G^{\#}$ to reflect the new $\hat{V}_l$, and restarting A* search. Therefore, M* will never return a suboptimal path as long as case 2 holds. For case 2 to hold, there must be at least one vertex v_l such that C_l is a strict subset of I. $G^{\#}$ can be modified at most $(n-1)*|G|$ times before all collision sets are equal to I. Therefore, case 2 can only hold for a finite number of modifications of $G^{\#}$. Since either case 1 or case 2 hold by hypothesis, after finite time only case 1 will hold. Since M* will always find $\pi^*(v_I, v_F)$ if only case 1 holds, and cannot find a suboptimal path if case 2 holds, M* will find $\pi^*(v_I, v_F)$ in finite time.

We will now show that case 1 or case 2 must always hold. We proceed by showing that we can always find a path $\pi(v_k, v_F)$ which costs no more than $\pi^*(v_k, v_F)$ by exhaustively searching the configuration space of the robots in C_k, while the robots in $\bar{C}_k$ obey $\phi^{\bar{C}_k}$.

We first note that, by the form of of (4), if $\Psi(\pi(v_k, v_l)) = \emptyset$, then the path taken by a subset of robots $\Omega \subset I$ must be a collision free path in Q^{Ω}. Therefore, if $\pi^*(v_k, v_F)$ exists, we can find an optimal collision free path $\pi^{*\Omega}(v_k^{\Omega}, v_F^{\Omega}) \subset Q^{\Omega}$ for any subset Ω of robots, where $\pi^{*\Omega}(v_k^{\Omega}, v_F^{\Omega})$ is not necessarily the path taken by the robots $r^i \in \Omega$ along $\pi^*(v_k, v_F)$. We can therefore construct a path $\pi'_k(v_k, v_F) = \pi^{*C_k}(v_k, v_F) \times \pi^{\phi^{C_k}}(v_k, v_F)$, which costs no more than $f(\pi^*(v_k, v_F))$.

$$f(\pi'_k(v_k, v_F)) = f^{C_k}(\pi^{*C_k}(v_k, v_F)) + \sum_{j \in \bar{C}_k} f^j(\pi^{\phi^j}(v_k, v_F)) \quad (11)$$

$$= \min_{\Psi(\pi^{C_k}(v_k, v_F))=\emptyset} f^{C_k}(\pi^{C_k}(v_k, v_F)) + \min \sum_{j \in \bar{C}_k} f^j(\pi^j(v_k, v_F)) \quad (12)$$

$$= \min_{\pi(v_k, v_F) \text{ s.t. } \Psi(\pi^{C_k}(v_k, v_F))=\emptyset} f(\pi(v_k, v_F)) \quad (13)$$

$$\leq \min_{\pi(v_k, v_F) \text{ s.t } \Psi(\pi^{C_l}(v_k, v_F))=\emptyset, C_k \subset C_l} f(\pi(v_k, v_F)) \quad (14)$$

$$\leq \min_{\pi(v_k, v_F) \text{ s.t.} \Psi(\pi(v_k, v_F))=\emptyset} f(\pi(v_k, v_F)) \quad (15)$$

$$\leq f(\pi^*(v_k, v_F)) \quad (16)$$

We know that the successor v_l of v_k along $\pi'_k(v_k, v_F)$ is in $\hat{V}_k$ by (7). Furthermore, we know that $C_l \subset C_k$ by (9), so by (14) and (15)

$$f(\pi'_k(v_k, v_l)) + f(\pi'_l(v_l, v_F)) \leq f(\pi'_k(v_k, v_F)) \leq f(\pi^*(v_k, v_F)) \quad (17)$$

Using the above two facts, we can construct a path $\pi''(v_I, v_F) \in G^{\#}$ which either satisfies case 1 or case 2. We construct $\pi''(v_I, v_F)$ by starting at v_I, and choosing the m'th vertex v_m in $\pi''(v_I, v_F)$ to be the neighbor of v_{m-1} on $\pi'(v_{m-1}, v_I)$. Applying (17) backwards from the last vertex from v_F to v_I guarantees that $f(\pi''(v_I, v_F)) \leq f(\pi'_I(v_I, v_F)) \leq f(\pi^*(v_I, v_F))$. If $\pi''(v_I, v_F) = \pi^*(v_I, v_F)$ then case 1 is satisfied. Otherwise, there is a vertex $v_k \in \pi''(v_I, v_F)$ such that $\Psi(v_k) \neq \emptyset$. By construction, $\Psi(v_k) \not\subset C_l$, where v_l is the predecessor of v_k. By (8), $f(\pi''(v_I, v_k)) + h(v_k) \leq f(\pi''(v_I, v_F)) \leq f(\pi^*(v_I, v_F))$, so case 2 is satisfied. [2] We have now shown that case 1 or case 2 must always hold. We can therefore conclude that M* will find $\pi^*(v_I, v_F)$, if it exists, in finite time. Since M* is guaranteed to find the optimal collision free path, or to determine that no valid path exists in finite time, M* is complete and optimal with respect to $f(\pi(.))$.

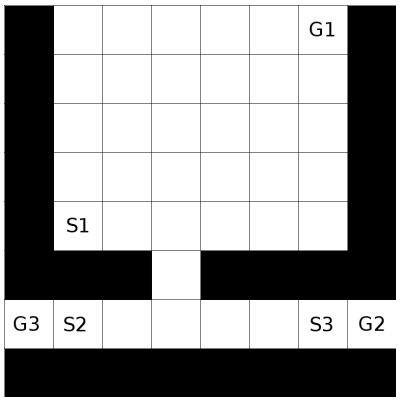

Fig. 4: In this example, 3 robots move from their initial positions S1, S2, and S3, to their goal positions, G1, G2, and G3. The world is a 4-connected grid, and the robots try to minimize the total distance traveled. Robot 1 has multiple optimal paths, but can safely be fully decoupled from robots 2 and 3. Therefore, only one of the optimal paths for robot 1 needs to be considered. A* is unable to recognize this decoupling, and for any joint path of robots 2 and 3, must consider all optimal paths for robot 1.

VI. Benefits Compared to A*

While the worst case computational cost of M* grows exponentially with the number of robots, as does the cost of A*, M* has two substantial advantages in the average case. First of all, unlike A*, M* does not need to add every neighbor to the open list. Doing so quickly becomes prohibitive for A*, as a vertex in a system of 13 robots on four-connected grids has over one billion neighbors. Secondly, A* must consider all regions of the configuration space for which $f(\pi^*(v_I, v_k)) + h(v_k) < f(\pi^*(v_I, v_F))$. M* can safely ignore such regions when they represent alternate paths for robots that aren't involved in collisions. Consider the case in Figure 4. Robot 1 can safely be decoupled from the planning for robots 2 and 3, but has multiple optimal paths. As a result, subdimensional expansion can safely constrain robot 1 to a single optimal path, and plan for robots 2 and 3 separately. A* cannot recognize this decoupling, and so must consider all optimal paths for robot 1 along with any joint paths for robots 2 and 3 it considers.

VII. Variants

A. Inflated M*

One problem with the basic M* implementation is that every time a new robot is found to be involved in a collision, it is added to the collision set of v_I. Unless $f(\pi^*(v_I, v_F)) = f(\pi^{\phi}(v_I, v_F))$, v_I must then be re-expanded at a computational cost that is exponential in the total number of

[2] If $\pi''(v_I, v_F)$ exits G', it may reenter G' at v_k such that $\Psi(v_k) \neq \emptyset$. In this case, v_k has no out neighbors so $\pi^*(v_I, v_F)$ will terminate at v_k and not reach v_F. However, the predecessor of v_k is not in G', so its collision set is $\emptyset \not\subset C_k$. Therefore, case 2 will hold

robots that have been found to collide. One improvement comes from the existing literature on A*. If the heuristic is multiplied by some $\epsilon > 1$, A* will find a path which costs no more than $\epsilon * f(\pi^*(v_I, v_F))$, and generally will find a path more quickly [2], [12], [20]. The logic in section V can be extended to show that M* has the same sub-optimality bound when used with an inflated heuristic.

The inflated heuristic biases the search towards the leaves of the search tree close to the goal, where a solution is more likely to be found quickly. In addition, these vertices will generally have a smaller collision set, resulting in a lower dimensional search space.

B. Recursive M*

Another area where M* can be improved is in the handling of multiple physically separated but simultaneously interacting sets of robots. Basic M* must couple the planning between all such sets of robots, even though they may have no mutual interaction. We can extend equations (11)-(15) from dealing with coupled planning for a single subset of robots to separately planning for multiple disjoint subsets of robots, $C_k, \ldots, C_o$. The resulting path, $\pi'(.) = \pi^{*C_k}(.) \times \ldots \times \pi^{*C_o}(.) \times \pi^{\phi I \setminus (C_k \bigcup \ldots \bigcup C_o)}(.)$, maintains the critical property, $f(\pi'(.)) \leq f(\pi^*(.))$. Therefore, we can find a path for each independent set of interacting robots, and use the resulting paths to constrain exploration in the same manner that individually optimal policies are used to constrain exploration for individual robots. Doing so results in worst case computational cost that is exponential in the size of the largest set of mutually colliding robots, instead of the total number of colliding robots. We term this variant *recursive M**, or *rM**.

Implementing recursive M* requires comparatively few modifications. First of all, the collision set now consists of the largest disjoint sets that can be formed from the collisions that can be reached from v_k. For example, if the collisions $\{1,2\}, \{2,3\}, \{4,5\}$ can be reached from v_k, then $C_k = \{\{1,2,3\}, \{4,5\}\}$. If r^i is not in any element of C_k, then it obeys ϕ^i. Otherwise, r^i follows the optimal path for the subset of robots $\hat{C} \in C_k$ to which it belongs. This path is found by recursively using rM* to find paths for these subproblems. The successor of v_k on the paths for each subset of robots in C_k are combined with $\phi^{\bar{C}_k}(v^{\bar{C}_k})$ to generate a single successor for v_k. The exception is if $C_k = I$, in which case $\hat{V}_k$ is computed as usual for M*, using $\hat{C}$ as the collision set. This functions as the base case of the recursive calls to rM*.

VIII. Results

We tested M* in simulations run on a Core i7 computer at 2.80 GHz with 12 Gb of RAM. All simulations were implemented in unoptimized python. We chose to use a square, four-connected grid with a density of 104 cells per robot as our workspace. Scaling the workspace with the number of robots kept the density of robots in the workspace constant, allowing us to vary the number of robots without also changing how crowded the robots were. Each cell in the workspace had an independent 35% chance of being marked as an obstacle. Initial and goal positions for each robot were chosen randomly, but were chosen such that there was always a path from the initial position of a robot to its goal position (Figure 5). Each robot incurred a cost of one when not in its individual goal state, and no cost when in its goal state. Each trial was given at most five minutes to find a solution. We tested 100 random envrionments for each number of robots.

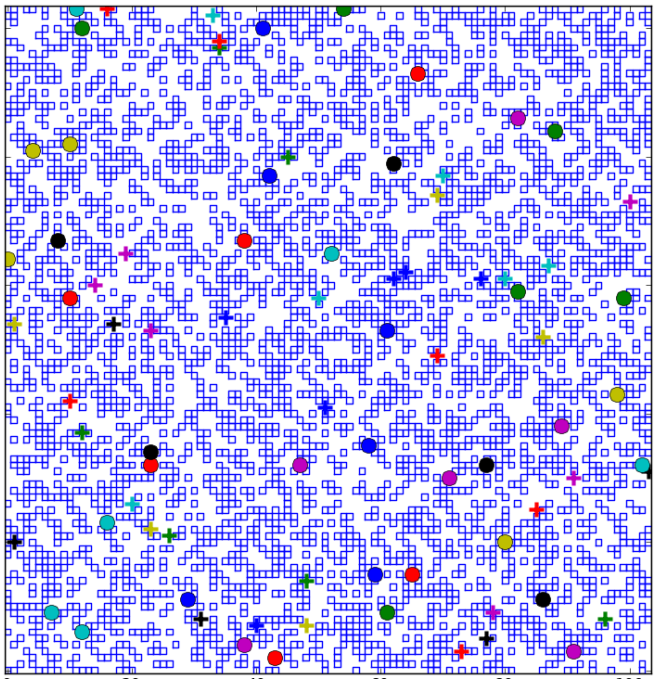

Fig. 5: A typical configuration for a 40 robot test run. Circles represent initial positions of the robots, squares represent obstacles, and crosses represent goal positions. We tested 100 such randomly generated environments for each number of robots.

We ran A*, M* and rM* with both the uninflated heuristic, and the heuristic inflated by a factor of 2. We recorded the percentage of trials which found a solution within five minutes, as well as the time required to find a solution by the 10'th, 50'th, and 90'th percentile of trials. Run times are plotted on semi-log plots, where exponential growth with the number of robots will appear as straight lines (Figure 6).

A* was unable to find any paths for problems involving seven or more robots, due to the cost of adding all of the neighbors of each expanded vertex to the open list. The time required to find solutions shows the expected exponential growth with the number of robots. M* and rM* both show performance substantially superior to that of A*. rM* has roughly three times the success rate of M* for the uninflated case at 10 robots, but shows even an even greater performance increase when the heuristic function is inflated. Using an inflated heuristic, rM* has run times of approximately one and a half orders of magnitude less than basic M* for systems of 20 robots, and scales to twice as many robots with reasonable success rates (Figure 6). Most importantly, the time to solution plots for inflated rM* are clearly sublinear on the logarithmic axis. This indicates that for the enviornments we investigated, the average case

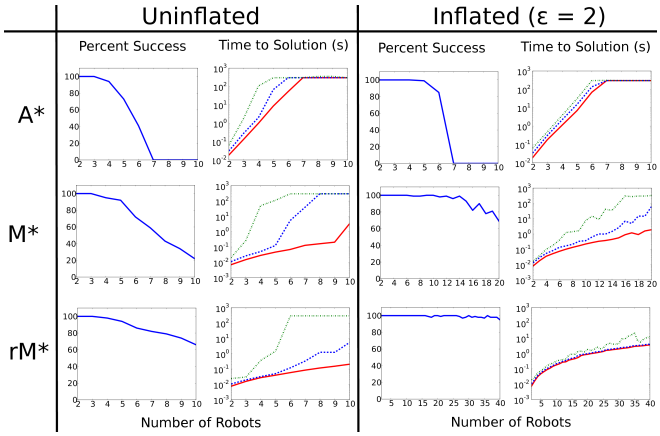

Fig. 6: We plot the percent of trials in which each algorithm was able to find a solution within 5 minutes, and the 10'th, 50'th, and 90'th percentile of times required to find a solution for A*, M*, and recursive M* with both uninflated and inflated heuristics. The plateauing that is apparent many of the time plots are the result of the algorithm timing out in increasingly large percentages of trials. We only simulated A* and inflated A* to 8 robots, because they always timed out for 7 or more robots. To allow A*, M*, and rM* to be plotted over similar domains, we assumed that A* and inflated A* would always timeout for systems of 9 and 10 robots. Inflated M* and inflated rM* were able to solve 20 and 40 robot problems respectively, so their plots reflect these domains.

computational cost of rM* grows sub-exponentially with the number of robots.

Increasing the time limit to 25 minutes, we were able to plan for 100 robots with a 83% success rate and a median time to solution of 54 seconds. However, M* is heavily memory limited, we cannot substantially increase the run times to handle tougher problems.

IX. Future Work

One weakness of M* is that a search will fail if a sufficient number of robots are concentrated at a single choke point, as this will force M* to search an excessively large space. The cost of expanding a node grows in a predictable manner, so it is comparatively easy to determine when a vertex has a collision set that is 'too big'. A possible solution for rM* would be to use a priority planner [6] instead of recursive calls to M* to generate the policy for sets of robots that are deemed 'too large'. While doing so would cause the loss of completeness and optimality guarantees, it may allow for a path to be found within the memory and time constraints when not otherwise possible. Signaling the user that optimality can not be guaranteed would be trivial in those cases when this approach is necessary.

Subdimensional expansion can be applied to path planning algorithms besides A*. We intend to explore using D* [22], and Anytime A* [17] as the planner for subdimensional expansion in discrete worlds. We will apply M* to more complex envrionments by using a PRM to generate the graph representing the individual robot configuration space [10]. Systems with kineodynamic constraints can be handled by implementing subdimensional expansion using RRTs [14] as the planner. This can be done by modifying how the RRT expands towards a sample. Only the robots in the collision set will actually move towards the coordinates specified by the sample. All other robots will obey their individual policy.

References

[1] N. Ayanian and V. Kumar. Decentralized feedback controllers for multi-agent teams in environments with obstacles. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1936–1941, May 2008.

[2] Blai Bonet and Hector Geffner. Planning as heuristic search. *Artificial Intelligence*, 129(1-2):5 – 33, 2001.

[3] S. Carpin and E. Pagello. On parallel RRTs for multi-robot systems. In *Proc. 8th Conf. Italian Association for Artificial Intelligence*, pages 834–841. Citeseer, 2002.

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

[5] C. M. Clark, S. M. Rock, and J. C. Latombe. Motion planning for multiple robot systems using dynamic networks. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 4222–4227, 2003.

[6] M. Erdmann and T. Lozano-Perez. On multiple moving objects. *Algorithmica*, 2(1):477–521, 1987.

[7] Robert W Ghrist and Daniel E Koditschek. Safe cooperative robot dynamics on graphs. *SIAM Journal on Control and Optimization*, 40(5), 2002.

[8] P.E. Hart, N. J. Nilsson, and B Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2), July 1968.

[9] K. Kant and S.W. Zucker. Toward efficient trajectory planning: The path-velocity decomposition. *The International Journal of Robotics Research*, 5(3):72, 1986.

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

[11] Jelle R. Kok, Pieter Jan 't Hoen, Bram Bakker, and Nikos Vlassis. Utile coordination: Learning interdependencies among cooperative agents. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, 2005.

[12] Richard E. Korf. Linear-space best-first search. *Artificial Intelligence*, 62(1):41 – 78, 1993.

[13] K. Madhava Krishna, Henry Hexmoor, and Srinivas Chellappa. Reactive navigation of multiple moving agents by collaborative resolution of conflicts. *Journal of Robotic Systems*, pages 249–269, 2005.

[14] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. In *Proceedings IEE International Conference on Robotics and Automation*, 1999.

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

[16] Stephane Leroy, Jean-Paul Laumond, and Thierry Siméon. Multiple path coordination for mobile robots: A geometric algorithm. In *IJCAI '99: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1118–1123, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[17] M. Likhachev, G. Gordon, and S. Thrun. ARA*: Anytime A* search with provable bounds on sub-optimality. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Proceedings of Conference on Neural Information Processing Systems (NIPS)*. MIT Press, 2003.

[18] Ryan Malcom. Multi-robot path-planning with subgraphs. In *Australasian Conference on Robotics and Automation*, 2006.

[19] Francisco S. Melo and Manuela Veloso. Learning of coordination: Exploiting sparse interactions in multiagent systems. In *'Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems*, May 2009.

[20] Judea Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Welsley, 1984.

[21] M. Saha and P. Isto. Multi-robot motion planning by incremental coordination. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 5960–5963, Oct. 2006.

[22] Anthony Stentz. Optimal and efficient path planning for unknown and dynamic environments. *International Journal of Robotics and Automation*, 10:89–100, 1993.

[23] Jur van den Berg, Jack Snoeyink, Ming Lin, and Dinesh Manocha. Centralized path planning for multiple robots: Optimal decoupling into sequential plans. In *Proc. Robotics: Science and Systems - RSS'09*, 2009.