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

Glenn Wagner and 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.

1 Introduction

Multirobot systems are attractive for surveillance, search and rescue, and warehouse automation applications. Unfortunately, the flexibility and redundancy that make multirobot systems appealing also make assigning robots to tasks and planning collision free paths to perform those tasks difficult. In this work, we describe a novel method, called *subdimensional expansion*, for efficiently generating collision free paths for multiple robots [17]. Subdimensional expansion initially assumes that a path can be found for each robot to the goal in the robots' individual configuration

Glenn Wagner
The Robotics Institute, Carnegie Mellon University, Newell Simon Hall 4221,
5000 Forbes Ave, Pittsburgh, PA 15213
e-mail: gswagner@andrew.cmu.edu

Howie Choset
The Robotics Institute, Carnegie Mellon University, Newell Simon Hall 3205,
5000 Forbes Ave, Pittsburgh, PA 15213
e-mail: choset@cs.cmu.edu

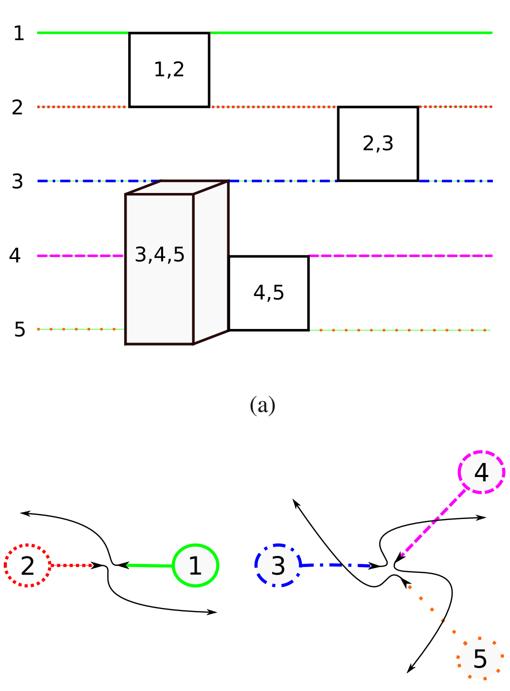

Fig. 1 A conceptual visualization of a variable dimensionality search space for five robots (a). Initially each robot is constrained to its individually optimal path, represented by a single line, but when robots 1 and 2 collide (b), the local dimensionality of the search space must be increased, as 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.

space, without coordinating with other robots. When the paths of multiple robots intersect, the joint configuration space of those robots is locally constructed and planning occurs in this joint space. Once coordination is no longer necessary, planning reverts to the low dimensional individual spaces until the goal is reached or another collision is found (Fig 1).

2 Prior Work

Multirobot path planning algorithms can be divided into two categories: coupled and decoupled [11]. A coupled algorithm seeks to find a path in the full configuration space of a system [1, 2, 6], which grows exponentially with the number of robots. 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 [5, 8, 12, 13, 15]. 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.

Ideally, a multirobot path planner would combine the scalability of decoupled approaches with the completeness and optimality guarantees of a coupled approach. Some prior work seeks to inherit the benefits of both approaches by dynamically fitting the degree of coupling to the specific problem to be solved. Krishna *et al.* developed an approach for decentralized dynamic coupling of robots for velocity planning [10]. 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 [4]. Joint plans are computed for groups of robots capable of mutual communication. Paths are re-planned 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.* [16] 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. 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.* [9] 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 [14] 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 only occurs when a robot learns to take the coordinate action.

3 Problem Statement

The objective of subdimensional expansion is to find an optimal collision free path for a set of n robots, $r^i, i \in I = \{1, \ldots, n\}$, in a common workspace $\mathscr{W}$. We denote the start configuration of each robot in its individual configuration space as $q^i_s \in Q^i$. The start configuration of all robots can be described as a point in the full configuration space $q_s \in Q = \prod_{i=1}^{n} Q^i$. The goal configuration for each robot is denoted $q^i_f \in Q^i$, while the joint goal configuration is denoted $q_f \in Q$. The optimal collision free path from q_s to q_f is denoted $\pi^*(q_s, q_f)$.

The cost $f(\pi)$ of the path π in the full configuration space is assumed to be the sum of the costs $f^i(\pi^i) \geq 0$ of the paths π^i of the individual robots, *i.e.*,

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

where $\pi(q_i, q_j)$ represents a path from q_i to q_j.

A collision function is defined to represent collisions between robots r^i and r^j,

$$\Psi^{ij}(q) = \begin{cases} \{i, j\}, & A(q^i) \cap A(q^j) \neq \emptyset \\ \emptyset, & \text{otherwise} \end{cases}. \tag{2}$$

where $A(q^i)$ is the subset of $\mathscr{W}$ 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.

$$\Psi(q) = \bigcup_{i \neq j \in I} \Psi^{ij}(q). \tag{3}$$

The collision function is "overloaded" to apply to paths, $\Psi(\pi(.)) = \bigcup_{q \in \pi(.)} \Psi(q)$.

4 Subdimensional Expansion

Multirobot systems which obey (1) and (3) have a natural decoupling between individual robots. (1) guarantees that no joint path can be cheaper than the path found by optimizing for each robot separately, which is thus a good starting point for multirobot path planning. The individual paths are combined to form a joint path for the entire system. When robot-robot collisions are found along the joint path, planning is locally coupled for the involved robots while uninvolved robots proceed independently, which is sufficient to guarantee that a path will be found due to the form of (3). Subdimensional expansion is a method for encoding the dynamic coupling into the geometry of the search space, thereby allowing conventional algorithms to search the necessary portions of the joint configuration space.

$Q^\#$ is the variably dimensional search space embedded in Q constructed by subdimensional expansion. Note that $Q^\#$ is dynamically constructed as a planner searches $Q^\#$. The simultaneous construction and search of $Q^\#$ continues until a path is found or determined to be impossible. The construction of $Q^\#$ is guided by information about robot-robot collisions found by the planner. Thus, the search space is tailored to the specific problem at hand, allowing the search of a low dimensional space while also guaranteeing the optimal path will be found.

4.1 Approach

Each robot starts with its own individually optimal policy $\phi^i : Q^i \to TQ^i$ which maps the position of a robot to its motion. ϕ^i is chosen such that the path induced

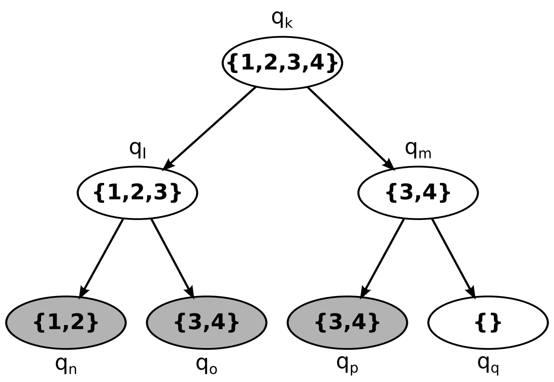

Fig. 2 Representation of a search tree and resultant collision sets. Ovals represent configurations in Q. Arrows represent searched path from the higher to the lower configuration. Gray ovals correspond to states with robot-robot collisions. The set contained inside the oval represents the collision 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 the planner has not found a path from q_q to any state with a collision, C_q is empty.

by obeying ϕ^i from any point $q_k^i \in Q^i$ is an optimal path to $q_f^i \in Q^i$, and is denoted $\pi^{\phi^i}(q_k^i, q_f^i)$. The joint path generated by all robots obeying their individually optimal policies is denoted $\pi^{\phi}(q_k, q_f) = \prod_{i \in I} \pi^{\phi^i}(q_k^i, q_f^i)$.

At each instant during the search, the planner initially takes the optimistic view that the individually optimal path from q_k will be collision free, without specific information to the contrary. The planner maintains 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

$$C_k = \bigcup_{\pi \in \pi(q_k)} \Psi(\pi) \tag{4}$$

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).

Initially, $Q^{\#}$ is $\pi^{\phi}(q_s, q_f)$, with each robot restricted to following its individually optimal path. When the planner finds a collision, it expands $Q^{\#}$ by locally allowing the robots involved in the collision to deviate from their individually optimal policies. Naturally, the set of robots not in the collision set, $\bar{C} = I \setminus C$ remain restricted to their individually optimal paths, in line with the optimistic belief that this portion of the path is collision free. These constraints are encoded in the geometry of the search space $Q^{\#}$ by proper choice of the tangent space $T_{q_k}Q^{\#}$ of $Q^{\#}$ at q_k.

$$T_{q_k}Q^{\#} = \mathbf{t}^{\bar{C}_k}(q_k) \times \prod_{j \in C_k} T_{q_k^j} Q^j \tag{5}$$

where $\mathbf{t}^{\bar{C}_k}(q_k) \in T_{q_k^{\bar{C}_k}} Q^{\bar{C}_k}$ is tangent to the joint individually optimal path for the robots in $\bar{C}_k$. $Q^\#$ is then grown differentially along $T_{q_k} Q^\#$ from q_k, expanding into a higher dimensional space when the collision set is large, or along the one-dimensional individual optimal path when the collision set is empty.

*4.2 M**

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^i_{kl} which represent valid transitions connecting $v^i_k \in V^i$ to $v^i_l \in V^i$. 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. The full configuration space of the system is represented by 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 $e^i_{kl} \in E^i$ and $e^j_{kl} \in E^j$. The vertex in G which represents the start configuration of the system is denoted v_s, 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 [7]. 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 $\pi(v_s, v_k)$ 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, from the open list is 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.

M* is similar to A*, however the expansion step is a little different: M* only considers the *limited neighbors* of v_k, a subset of the neighbors of v_k in G, determined by C_k, providing the benefit of only exploring the "necessary" subspace of the configuration space. 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 $\phi^i(v^i_k)$, where v^i_k 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^j_k in Q^j

$$\hat{V}_k = \left\{ v_l | \forall i \in I,\ v^i_l\ s.t. \begin{cases} e^i_{kl} \in E^i, & i \in C_k \\ v^i_l = \phi^i(v^i_k), & i \notin C_k \end{cases} \right\} \tag{6}$$

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

Information about collisions must be passed back along all searched paths that reach them. To this end, the planner maintains a *backpropagation set* for each vertex v_k, which is the set of all vertices v_l that were expanded while v_k was in $\hat{V}_l$. The

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)
```

backpropagation set is thus the set of neighbors of v_k through which the planner has found a path to v_k. The planner propagates 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. The planner then adds C_l to the collision set of each vertex in the backpropagation set of v_l, and repeats 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 be added back to the open list so that these new paths can be searched (See Algorithm 1).

Finally, since $f(\pi^\phi(v_k,v_f))$ is a lower bound on the cost of all paths $\pi(v_k,v_f)$, we use it as the heuristic function for M*. Denote the heuristic function

$$h(v_k) = f(\pi^\phi(v_k,v_f)) \leq f(\pi^*(v_k,v_f)). \tag{7}$$

M* is described in algorithm 2.

We developed two variants of M*, inflated M* and recursive M* (rM*). Inflated M* multiplies the heuristic by inflation factor $\varepsilon > 1$ to find a suboptimal path quickly. rM* is a hierarchical planner that breaks the path planning problem into multiple sub-problems, by separating the planning for non-interacting groups of mutually interfering robots. A sub-planner is recursively generated for each such group to find a path for the colliding robots to the goal. Sub-planners continue to be generated for smaller groups of colliding robots until the collision involves every robot handled by the sub-planner.

4.3 Graph-Centric Description

The description of M* in 4.2 provides a local description of the search process, which is useful for implementation. However, the local description makes proving the global properties of M* difficult. We now present an alternative description of M* which better captures the global properties, but is not appropriate for implementation.

When examining algorithm 2 we see that M* differs from A* in the existence of the **backprop** function, and the neighbors added to the open list during the

Algorithm 2. Pseudocode for M*

```
for all $v_k \in V$ do
    $v_k$.cost $\leftarrow$ MAXCOST
    $C_k \leftarrow \emptyset$
$v_s$.cost $\leftarrow 0$
$v_s$.back_ptr $= \emptyset$
open = $\{v_s\}$
while True do
    open.sort() {Sort in ascending order by v.cost + h(v)}
    $v_k$ = open.pop(0)
    if $v_k = v_s$ 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
```

expansion of a vertex. 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 new robot-robot collision is found. By thinking of M* as alternating between running A* on $G^{\#}$ and updating $G^{\#}$ based on partial results, we can exploit the optimality and completeness 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.

$G' = \{V', E'\}$ represents the portion of G which has been searched by M*. 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 explored by the planner, we can use G' to define the collision set

$$C_k = \begin{cases} \Psi(v_k) \bigcup_{v_l \text{ s.t. } \exists \pi(v_k, v_l) \subset G'} \Psi(v_l) & v_k \in G' \\ \emptyset & v_k \notin G' \end{cases} \tag{8}$$

If $v_k \notin G'$, then M* has never visited v_k. Thus in accordance with the optimistic view that v_k and $\pi^\phi(v_k, v_f)$ are collision free, C_k is set to the empty set until v_k is added to the open list.

$\hat{G}_k$ represents the portion of the graph which will be explored when v_k is expanded, and 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 $v_k \in \hat{G}$ 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'.

$G^\#$ can now be defined as

$$G^\# = G' \bigcup_{v_k \in G'} \left(\hat{G}_k \bigcup_{v_l \in \hat{G}_k \setminus G'} G_l^\phi \right) \tag{9}$$

As a result of the 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.

4.4 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 [3]. M* will be shown to be both complete and will find a minimal cost path. As demonstrated in 4.3, M* can be treated as alternating between running A* search on $G^\#$ and modifying $G^\#$ based on the partial search results. Since A* is complete and cost optimal [7], M* is complete and cost optimal if $G^\#$ will contain $\pi^*(v_s, v_f)$ after a finite number of modifications or, if $\pi^*(v_s, v_f)$ does not exist, $G^\#$ will be modified at most a finite number of times.

We start by assuming that no solution exists, and show that M* will terminate in finite time without returning a path. $G^\#$ is only modified when the collision set of at least one vertex in $G^\#$ is modified. Each modification adds one or more robot to the collision set, thus each collision set can be modified at most $n-1$ times, as the first modification must add at least two robots. Therefore, $G^\#$ can be modified at most $(n-1) * |G|$ times. A* will expand each vertex in a graph at most once [7]. Thus, M* will always terminate in finite time.

M* will never return a path containing a robot-robot collision. A vertex v_k at which a robot-robot collision takes place can only have outneighbors if $v_k \notin G'$. However, before a path through v_k can be returned, v_k must be added to G', which will trigger a modification of $G^\#$ that removes the outneighbors of v_k. Therefore, M* will never return a path containing a robot-robot collision. Thus M* will correctly determine that no valid path exists in finite time.

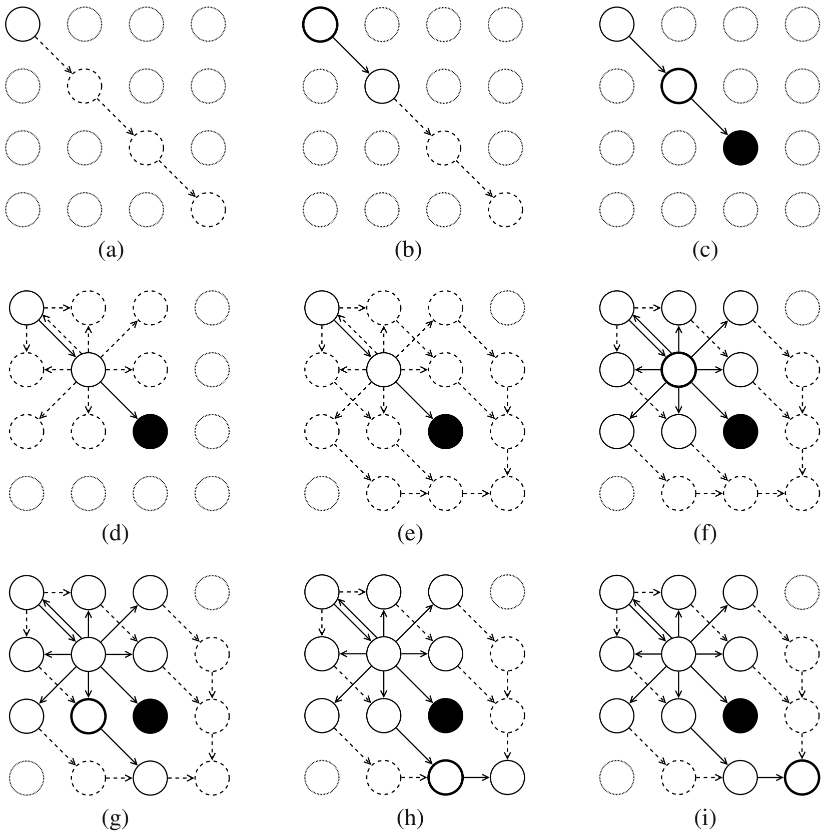

Fig. 3 The above figure shows how G' and $G^{\#}$ evolve in the configuration space of two one-dimensional robots. Vertices are represented as circles, with arrows representing directed edges. G' is denoted by solid lines, while $G^{\#} \backslash G'$ is shown as dashed lines. $G \backslash G^{\#}$ is 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_s 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.

Next, assume that a path from v_s to v_f exists. We will show that M* will find $\pi^*(v_s,v_f)$ as long as one of two cases is always true. We will then prove that one of these two cases must always hold. Assume that $G^\#$ always contains either

Case 1: an optimal, collision free path, $\pi^*(v_s,v_f)$, or
Case 2: a path $\pi(v_s,v_c)$ s.t. $f(\pi(v_s,v_c))+h(v_c) \leq f(\pi^*(v_s,v_f))$, and $\exists\, v_d \in \pi(v_s,v_c)$ s.t. $\Psi(v_c) \not\subset C_d$

If case 1 holds, running A* on $G^\#$ will find $\pi^*(v_s,v_f)$, unless there exists a cheaper path $\tilde{\pi}(v_s,v_f) \subset G^\#$. By the definition of $\pi^*(.)$, there must be a vertex $v_k \in \tilde{\pi}(v_s,v_f)\ s.t.\ \Psi(v_k) \neq \emptyset$, and by (7) $f(\tilde{\pi}(v_s,v_k))+h(v_k) < f(\pi^*(v_s,v_f))$. $v_k \notin G'$, as otherwise it would have no outneighbors, which implies $C_k = \emptyset$. As a result, v_k fulfills the roles of both v_c and v_d in the definition of case 2. Therefore M* will find $\pi^*(v_s,v_f)$ if case 1 holds, unless case 2 also holds.

If case 2 holds, then v_c will be added to G' before A* finds any path to v_f that costs more than $f(\pi^*(v_s,v_f))$ [7]. Adding v_c to G' will modify C_d, which will in turn change $G^\#$ to reflect the new $\hat{V}_d$ and restart 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_d such that C_d 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 to $G^\#$. By hypothesis, either case 1 or case 2 holds, which implies that within finite time only case 1 will be true. M* will thus find $\pi^*(v_s,v_f)$ 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 $\pi'(v_k,v_f)$, $f(\pi(v_k,v_f)) \leq f(\pi^*(v_k,v_f))$ under the restriction that the robots in $\bar{C}_k$ obey their individually optimal policies.

First note that, by the form of (3), if $\pi^*(v_k,v_f)$ exists, then for any subset of robots, $\Omega \subset I$, there exists an optimal, collision free path $\pi^{*\Omega}(v_k^\Omega,v_f^\Omega)$, which may not be the same as the paths taken by the robots in Ω in $\pi^*(v_k,v_f)$. Therefore a path $\pi'(v_k,v_f) = \pi^{*C_k}(v_k^{C_k},v_f^{C_k}) \times \pi^{\phi^{\bar{C}_k}}(v_k^{\bar{C}_k},v_f^{\bar{C}_k})$ can be constructed which costs no more than $f(\pi^*(v_k,v_f))$.

$$f(\pi'(v_k,v_f)) = f^{C_k}(\pi^{*C_k}(v_k^{C_k},v_f^{C_k})) + \sum_{j\in\bar{C}_k} f^j(\pi^{\phi^j}(v_k^j,v_f^j)) \tag{10}$$

$$= \min_{\pi^{C_k}(v_k^{C_k},v_f^{C_k})\ \text{s.t.}\ \Psi(\pi^{C_k}(v_k^{C_k},v_f^{C_k}))=\emptyset} f^{C_k}(\pi^{C_k}(v_k^{C_k},v_f^{C_k})) + \min \sum_{j\in\bar{C}_k} f^j(\pi^j(v_k^j,v_f^j)) \tag{11}$$

$$= \min_{\pi(v_k,v_f)\ \text{s.t.}\ \Psi(\pi^{C_k}(v_k^{C_k},v_f^{C_k}))=\emptyset} f(\pi(v_k,v_f)) \tag{12}$$

$$\leq \min_{\pi(v_k,v_f)\ \text{s.t}\ \Psi(\pi^{C_l}(v_k^{C_l},v_f^{C_l}))=\emptyset, C_k\subset C_l} f(\pi(v_k,v_f)) \tag{13}$$

$$\leq \min_{\pi(v_k,v_f)\ \text{s.t.}\Psi(\pi(v_k,v_f))=\emptyset} f(\pi(v_k,v_f)) \tag{14}$$

$$\leq f(\pi^*(v_k,v_f)) \tag{15}$$

The successor v_l of v_k along $\pi'(v_k, v_f)$ is in $\hat{V}_k$ by (6). Furthermore, $C_l \subset C_k$ by (8), so by (13) and (14)

$$f(\pi'(v_k, v_l)) + f(\pi'(v_l, v_f)) \leq f(\pi'(v_k, v_f)) \leq f(\pi^*(v_k, v_f)) \quad (16)$$

Using the above two facts, a path $\pi''(v_s, v_f) \in G^\#$ can be constructed which satisfies case 1 or case 2. Starting from v_s, the successor of the m'th vertex $v_m \in \pi''(v_s, v_f)$ is the successor of v_m in $\pi'(v_m, v_f)$. Applying (16) backwards from the last vertex from v_f to v_s guarantees that $f(\pi''(v_s, v_f)) \leq f(\pi_l'(v_s, v_f)) \leq f(\pi^*(v_s, v_f))$. If $\pi''(v_s, v_f) = \pi^*(v_s, v_f)$ then case 1 is satisfied. Otherwise, there is a vertex $v_k \in \pi''(v_s, v_f)$ 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 (7), $f(\pi''(v_s, v_k)) + h(v_k) \leq f(\pi''(v_s, v_f)) \leq f(\pi^*(v_s, v_f))$, so case 2 is satisfied. It has now been shown that case 1 or case 2 must always hold. Therefore M* will find $\pi^*(v_s, v_f)$, 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 $f(\pi(.))$.

5 Results

We tested the path planning performance of M* and its variants with randomly assigned goals. Our simulations were run on a Core i7 processor at 2.8 GHz with 12 Gb of RAM. All simulations are implemented in unoptimized Python. We chose a square, four-connected grid with a density of 104 cells per robot as our workspace, allowing the number of robots to vary without changing the level of congestion. Each cell in the workspace has an independent 35% chance of being an obstacle. Start and goal positions for each robot are chosen randomly, but such that a path always exists from the start position of a robot to its goal position (Figure 4). Each robot incurred a cost of one for each time step for which it was not at its goal. We tested 100 random environments for a given number of robots, and each trial was given at most five minutes to find a solution.

The time required to find solutions using A* shows the expected exponential growth with the number of robots. M* and rM* show performance substantially superior to A* ,which was unsuccessful for problems involving more than 6 robots. rM* has roughly three times the success rate of M* for the non-inflated case at 10 robots. Using an inflated heuristic, rM* has a greater performance increase, with run times of approximately one and a half orders of magnitude less than basic M* for systems of 20 robots, and scaling to twice as many robots with reasonable success rates (Figure 5). Most importantly, the time to solution plots for inflated rM* are sublinear on a logarithmic axis. This indicates that for the environments we investigated, the average computational cost of rM* grows sub-exponentially with the number of robots.

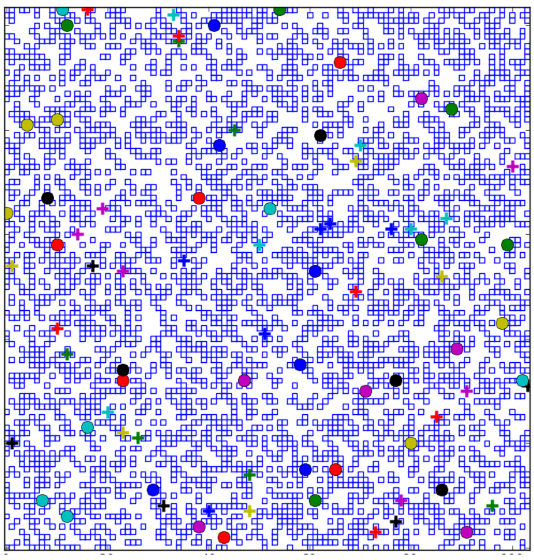

Fig. 4 A typical configuration for a 40 robot test run. Circles represent start positions of the robots, squares represent obstacles, and crosses represent goal positions. We tested 100 such randomly generated environments for each number of robots.

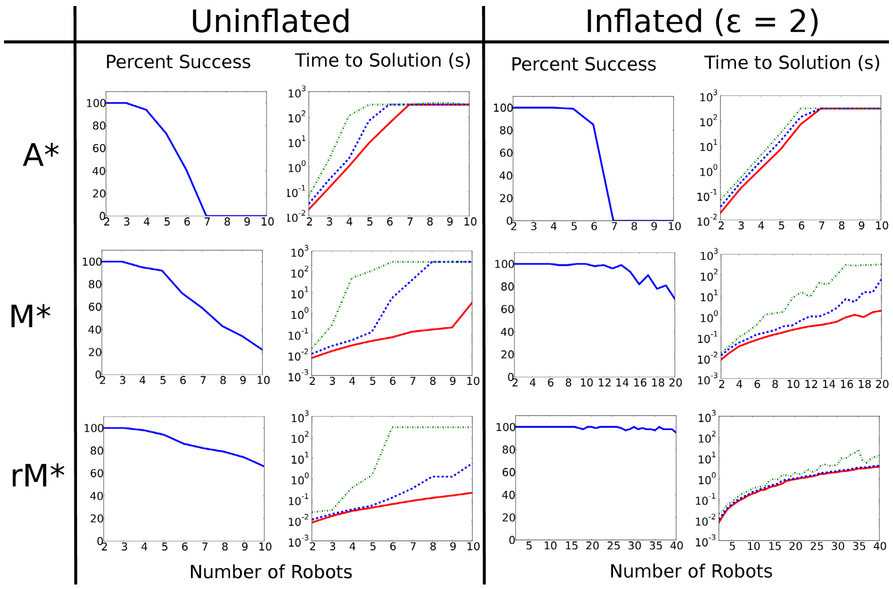

Fig. 5 We plot the percent of trials in which each algorithm was able to find a solution within 5 minutes, and the 10'th, 50'th, and 90'th percentile of times required to find a solution for A*, M*, and recursive M* with both non-inflated and inflated heuristics. The time to solution plots flatten out when a sufficient number of trials are halted by the time limit. A* and inflated A* are only simulated 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 time out for systems of 9 and 10 robots. Inflated M* and inflated rM* were able to solve 20 and 40 robot problems respectively, which is reflected in the domains of their plots.

6 Conclusions

We present a general approach to multirobot path planning, called subdimensional expansion, and an implementation for graph search, called M*. We demonstrate that M* can scale to problems involving large numbers of robots, while maintaining completeness and bounded suboptimality. These results illustrate the advantage of tailoring the search space to the individual problem being solved.

One weakness of M* is that 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 high dimensional space. One possible solution is to couple the path planning problem with the task assignment problem, and to avoid task assignments which require passing through said choke points. Our preliminary results indicate that such coupling can dramatically reduce the time required to find solutions, as well as reduce the cost of the resultant path compared to the optimal path for a task assignment found without considering robot-robot interactions.

Subdimensional expansion can be applied to path planning algorithms besides A*. In general, subdimensional expansion can be applied to nearly any path planning algorithm that produces a search tree which can be used to define collision sets. In particular, subdimensional expansion can be applied to RRTs and PRMs [18].

References

1. Ayanian, N., Kumar, V.: Decentralized feedback controllers for multi-agent teams in environments with obstacles. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 1936–1941 (May 2008)
2. Carpin, S., Pagello, E.: On parallel RRTs for multi-robot systems. In: Proceedings of 8th Conference Italian Association for Artificial Intelligence, pp. 834–841. Citeseer (2002)
3. Choset, H.M.: Principles of robot motion: theory, algorithms, and implementation. The MIT Press (2005)
4. Clark, C.M., Rock, S.M., Latombe, J.C.: Motion planning for multiple robot systems using dynamic networks. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 4222–4227 (2003)
5. Erdmann, M., Lozano-Perez, T.: On multiple moving objects. Algorithmica 2(1), 477–521 (1987)
6. Ghrist, R.W., Koditschek, D.E.: Safe cooperative robot dynamics on graphs. SIAM Journal on Control and Optimization 40(5) (2002)
7. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics 4(2) (July 1968)
8. Kant, K., Zucker, S.W.: Toward efficient trajectory planning: The path-velocity decomposition. The International Journal of Robotics Research 5(3), 72 (1986)
9. Kok, J.R., Hoen, P.J., Bakker, B., Vlassis, N.: Utile coordination: Learning interdependencies among cooperative agents. In: Proceedings of the IEEE Symposium on Computational Intelligence and Games (2005)
10. Krishna, K.M., Hexmoor, H., Chellappa, S.: Reactive navigation of multiple moving agents by collaborative resolution of conflicts. Journal of Robotic Systems, 249–269 (2005)

11. LaValle, S.M.: Planning algorithms. Cambridge Univ. Pr. (2006)
12. Leroy, S., Laumond, J.-P., Siméon, T.: Multiple path coordination for mobile robots: A geometric algorithm. In: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, pp. 1118–1123. Morgan Kaufmann Publishers Inc., San Francisco (1999)
13. Malcom, R.: Multi-robot path-planning with subgraphs. In: Australasian Conference on Robotics and Automation (2006)
14. Melo, F.S., Veloso, M.: Learning of coordination: Exploiting sparse interactions in multiagent systems. In: Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (May 2009)
15. Saha, M., Isto, P.: Multi-robot motion planning by incremental coordination. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5960–5963 (October 2006)
16. Jur van den Berg, J., Snoeyink, M., Lin, D.: Manocha. Centralized path planning for multiple robots: Optimal decoupling into sequential plans. In: Proceedings of Robotics: Science and Systems (2009)
17. Wagner, G., Choset, H.: M*: A complete multirobot path planning algorithm with performance bounds. In: Proceedings of IEEE International Conference on Intelligent Robots and Systems 2011 (September 2011)
18. Wagner, G., Kang, M., Choset, H.: Probabilistic path planning for multiple robots with subdimensional expansion. In: Proceedings of IEEE/RSJ International Conference on Robotics and Automation (May 2012)