Proceedings of the 1996 IEEE
International Conference on Robotics and Automation
Minneapolis, Minnesota - April 1996

Optimal Motion Planning for Multiple Robots Having Independent Goals

Steven M. LaValle
Dept. of Computer Science
Stanford University
Stanford, CA 94305
lavalle@robotics.stanford.edu

Seth A. Hutchinson
Dept. of Elect. & Comp. Engineering
University of Illinois
Urbana, IL 61801
seth@uiuc.edu

Abstract

This work makes two contributions to geometric motion planning for multiple robots: i) motion plans can be determined that simultaneously optimize an independent performance criterion for each robot; ii) a general spectrum is defined between decoupled and centralized planning.

By considering independent performance criteria, we introduce a form of optimality that is consistent with concepts from multi-objective optimization and game theory research. Previous multiple-robot motion planning approaches that consider optimality combine individual criteria into a single criterion. As a result, these methods can fail to find many potentially useful motion plans. We present implemented, multiple-robot motion planning algorithms that are derived from the principle of optimality, for three problem classes along the spectrum between centralized and decoupled planning: i) coordination along fixed, independent paths; ii) coordination along independent roadmaps; iii) general, unconstrained motion planning. Several computed examples are presented for all three problem classes that illustrate the concepts and algorithms.

1 Introduction

As robot applications continue to increase in complexity, the need for coordinating the efforts of multiple robots continues to expand. This paper addresses problems in which the task is to simultaneously bring each of two or more robots from an initial configuration to a goal configuration. In addition to ensuring collision avoidance, each robot has an independent objective to be optimized.

This final point differs from previous approaches to multiple-robot motion planning. Typically, if optimality is considered, performance measures for the individual robots are combined into a single scalar objective. For instance, in [5, 11] the objective is to minimize the time taken by the last robot to reach the goal. In [12], the performance measures are aggregated to yield a single objective. When individual objectives are combined, certain information about potential solutions and alternatives is lost (for general discussions, see [7, 10, 13]). For a given a vector of independent objective functionals, we show that there exists a natural partial ordering on the space of motion plans, yielding a search for the set of *minimal* motion plans.

In addition to introducing multiple-objective optimality to the multi-robot geometric motion planning, we expand the traditional view of centralized and decoupled planning by considering these two approaches as opposite ends of a spectrum. An approach that weakly constrains the robot motions before considering interactions between robots could be considered as lying somewhere in the middle of the spectrum. By utilizing this view, we show that many useful solutions can be obtained by constraining the robots to lie on independent, configuration space roadmaps. A roadmap is a one-dimensional network of curves, which is practical to utilize for our context since several general methods exist that produce them.

2 General Concepts

We first introduce some common geometric motion planning terminology. We consider each robot, $\mathcal{A}_i$, as a rigid object, capable of moving in a workspace that contains static obstacles. The position and orientation of the robot in the workspace are specified parametrically, by a point in an n-dimensional *configuration space*, $\mathcal{C}^i$. The *free configuration space*, $\mathcal{C}_{free}$ represents the open set of configurations in which $\mathcal{A}_i$ does not collide with an obstacle. In this work, we allow a robot to move in $\mathcal{C}_{valid}$, which is the closure of $\mathcal{C}_{free}$ (i.e., includes the boundary of $\mathcal{C}^i_{free}$).

We next define a *state space*, X, that simultaneously represents the configurations of all of the robots:

$$X = X^1 \times X^2 \times \cdots \times X^N, \tag{1}$$

in which each $x^i \in X^i$ represents the configuration of $\mathcal{A}_i$. We can take $X^i = \mathcal{C}^i_{valid}$, or further restrict X^i to be the image of a path or roadmap, as discussed in the coming sections. We use the notation $\mathcal{A}_i(x^i)$ to refer to the transformed robot, $\mathcal{A}_i$, at configuration x^i.

Let $\mathcal{A}_i^\circ$ denote the interior of $\mathcal{A}_i$ (i.e., the open set corresponding to the exclusion of the boundary of $\mathcal{A}_i$). We define (see Figure 1)

$$X^{ij}_{coll} = \{x \in X \mid \mathcal{A}_i^\circ(x^i) \cap \mathcal{A}_j^\circ(x^j) \neq \emptyset\}, \tag{2}$$

which represents the set of states in which robots i and j collide. The implication of using the interior of $\mathcal{A}_i$ is that we allow the robots to "touch". The *collision subset*, $X_{coll} \subset X$ is represented as the open set,

$$X_{coll} = \bigcup_{i \neq j} X^{ij}_{coll}. \tag{3}$$

A state is in the collision subset if the interior of two or more robots intersect. We define X_{valid} as the closed set, $X - X_{coll}$. Figure 1 shows the cylindrical structure that X_{coll} retains, which is exploited in our algorithms to significantly reduce the number of collision detections.

We consider a *state trajectory* as a continuous mapping $x : [0, T] \to X$. A trajectory for an individual robot is represented as $x^i : [0, T] \to X^i$ (a final time, T, is presented only to prevent the consideration of limits).

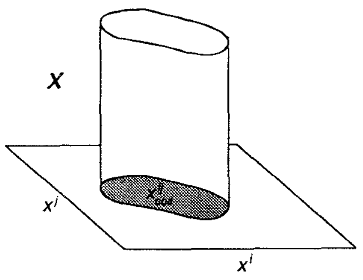

Figure 1. The set X^{ij}_{coll} and its cylindrical structure in X.

The motion of an individual robot, $\mathcal{A}_i$, is specified through the *state transition equation,*

$$\dot{x}^i(t) = f^i(x^i(t), u^i(t)) \tag{4}$$

in which $u^i(t)$ represents a *control function* for $\mathcal{A}_i$, which is chosen from a set of allowable controls.

We assume that the dynamics are negligible, and a robot is capable of switching between a fixed, maximum speed, $\|v^i\|$, and remaining motionless (this represents a typical assumption in geometric motion planning [2, 6, 8]). We next express the performance criteria for the robots. For each robot, $\mathcal{A}_i$, we define a *loss functional* of the form
$L^i(x_{init}, x_{goal}, u^1, \ldots, u^N) =$

$$\int_0^T g^i(t, x^i(t), u^i(t))dt + \sum_{j \neq i} c^{ij}(x(\cdot)) + q^i(x^i(T)), \tag{5}$$

which maps to the extended reals, and $c^{ij}(x(\cdot)) = \infty$ if $x(t) \subseteq X_{valid}$, and zero otherwise. Also, $q^i(x^i(T)) = 0$ if $x^i(T) = x^i_{goal}$, and ∞ otherwise. The function g^i represents a continuous cost function, which is a standard form that is used in optimal control theory. We additionally require, however, that

$$g^i(t, x^i(t), u^i(t)) = 0 \quad \text{if } x^i(t) = x^i_{goal}. \tag{6}$$

This implies that no additional cost is received while robot $\mathcal{A}_i$ "waits" at x^i_{goal} until time T. The middle term in (5), penalizes collisions between the robots. The function $q^i(x^i(T))$ in (5) represents the goal in terms of performance. If a robot, $\mathcal{A}_i$, fails to achieve its goal x^i_{goal}, then it receives infinite loss.

2.1 A Proposed Solution Concept

Suppose that a coordination problem has been posed in which the state space, X, is defined, along with initial and goal states, x_{init} and x_{goal}. We will use the notation γ^i to refer to a *robot strategy* for $\mathcal{A}_i$, which represents a possible choice of control function that incorporates state feedback, represented as $u^i(t) = \gamma^i(x, t)$. We refer to $\gamma = \{\gamma^1, \gamma^2, \ldots, \gamma^N\}$ as a *strategy.* Let Γ denote the set of all allowable strategies.

A *stationary strategy* is a special form of strategy that depends only on state, and not on a particular time. For the motion planning problems that we consider, the solutions are naturally stationary. If $T = \infty$, and f^i and g^i are time invariant, then the resulting solution strategies will be stationary. This is true since the objectives (5), and the effects of the control $u^i(t)$ on the system, remain invariant through the passage of time. The algorithms that we present in Sections 3-5 can be extended to handle time-varying nonstationary problems.

For a given x_{init} and strategy γ, the entire trajectory, $x(t)$, can be determined. If we assume that x_{init} and x_{goal} are given, then we can write $L^i(\gamma)$ instead of $L^i(x_{init}, x_{goal}, u^1, \ldots, u^N)$. Unless otherwise stated, we assume in the remainder of the paper that $L^i(\gamma)$ refers to the loss associated with implementing γ, to bring the robot from some fixed x_{init} to x_{goal}. Hence, we can consider the loss functional as a function on Γ.

In general, there will be many strategies in Γ that produce equivalent losses. Therefore, we define an equivalence relation, $\sim_L$, on all pairs of strategies in Γ. We say that $\gamma \sim_L \gamma'$ iff $L^i(\gamma) = L^i(\gamma')$ $\forall i$ (i.e., γ and γ' are equivalent). The equivalence relation, $\sim_L$, induces a partition of Γ into classes that produce equivalent losses. We denote the *quotient strategy space* by $\Gamma/\!\sim$, whose elements are the induced equivalence classes. An element of $\Gamma/\!\sim$ will be termed a *quotient strategy* and will be denoted as $[\gamma]_L$, indicating the equivalence class that contains γ.

We define a partial ordering, $\preceq$, on the space $\Gamma/\!\sim$. The minimal elements with respect to $\Gamma/\!\sim$ will be considered as the solutions to our problem. For a pair of elements $[\gamma]_L, [\gamma']_L \in \Gamma/\!\sim$ we declare that $[\gamma]_L \preceq [\gamma']_L$ if $L^i(\gamma) \leq L^i(\gamma')$ for each i. If it further holds that $L^j(\gamma) < L^j(\gamma')$ for some j, we say that $[\gamma]_L$ is *better* than $[\gamma']_L$. Two quotient strategies, $[\gamma]_L$ and $[\gamma']_L$, are *incomparable* if there exists some i, j such that $L^i(\gamma) < L^i(\gamma')$ and $L^j(\gamma) > L^j(\gamma')$. Hence, we can consider $[\gamma]_L$ to be either better than, *worse* than, equivalent to, or incomparable to $[\gamma']_L$. We can also apply the terms *worse* and *better* to representative strategies of different quotient strategies; for instance we could say that γ is better than γ' if $[\gamma]_L \preceq [\gamma']_L$. We say that $[\gamma^*]_L$ is a *minimal* strategy if for all $[\gamma]_L \neq [\gamma^*]_L$ such that $[\gamma]_L$ and $[\gamma^*]_L$ are not incomparable, we have $[\gamma^*]_L \preceq [\gamma]_L$.

2.2 Related Forms of Optimality

In this section we briefly state how the minimal strategies relate to optimality concepts from multiobjective optimization and dynamic game theory. See [7] for a more detailed discussion. The minimal quotient strategies are equivalent to the *nondominated* strategies used in multiobjective optimization and *Pareto optimal* strategies used in cooperative game theory. Furthermore, it can be shown that the minimal strategies satisfy the Nash equilibrium condition from noncooperative game theory, which implies that for a strategy $\gamma^* = \{\gamma^{1*} \ldots \gamma^{N*}\}$, the following holds for each i and each $\gamma^i \in \Gamma^i$:

$$L^i(\gamma^{1*}, \ldots, \gamma^{i*}, \ldots \gamma^{N*}) \leq L^i(\gamma^{1*}, \ldots, \gamma^{i}, \ldots \gamma^{N*}). \tag{7}$$

We can also consider the relationship between our minimal strategies, and scalar optimization. In multiobjective optimization literature, this is referred to as *scalarization* [10], in which a mapping that projects the loss vector to a scalar, while guaranteeing that optimizing the scalar loss produces a nondominated strategy. This is advantageous since standard optimization techniques can be applied to produce a minimal strategy. The tradeoff, however, is that the scalarizing function selects only a single minimal strategy. This makes the particular choice of a scalarizing function crucial, and information about the solution alternatives is lost.

We present a linear scalarizing function for which we have shown that optimizing the scalar objective yields a minimal strategy. This function is used in Section 5, in an algorithm that determines minimal solutions. Consider a vector of positive, real-valued constants, $\beta = [\beta_1 \ \beta_2 \ \ldots \ \beta_N]$, such that $\|\beta\| = 1$. If we take $\beta_i = \frac{1}{N}$ for all $i \in \{1, \ldots, N\}$, then the scalarizing function produces the average loss among the robots. In principle, this scalarizing function could be considered as a flexible form of prioritization. It has been shown that for a fixed β, if γ^* is a strategy that minimizes $H(\gamma, \beta)$, then the quotient strategy, $[\gamma^*]_L$ is minimal [7].

3 Motion Planning Along Fixed Paths

In this section we consider the problem of coordinating the motions of multiple robots, when each robot is independently constrained to traverse a fixed path. This work makes new contributions to the problem of coordinating multiple robots along fixed paths. First, we generalize the coordination space to more than two robots by exploiting the cylindrical structure of X_{coll}. The principle of optimality is then applied to yield an algorithm that determines all minimal quotient strategies. This algorithm can be specialized through scalarization to a standard dynamic programming algorithm, in which A^* search could be used to guide computation.

We assume that each robot, $\mathcal{A}_i$, is given a path, τ^i, which is a continuous mapping $[0,1] \rightarrow C^i_{valid}$. Without loss of generality, assume that the parameterization of τ^i is of constant speed. Let $\mathcal{S}^i = [0,1]$ denote the set of parameter values that place the robot along the path τ^i. We define a *path coordination space* as $\mathcal{S} = \mathcal{S}^1 \times \mathcal{S}^2 \times \cdots \times \mathcal{S}^N$.

A strategy $\gamma \in \Gamma$ must be provided in which $s_{init} = (0, 0, \ldots, 0)$ and $s_{goal} = (1, 1, \ldots, 1)$, and the robots do not collide. This corresponds to moving each robot from $\tau^i(0)$ to $\tau^i(1)$, and we assume that a robot, $\mathcal{A}_i$, monotonically moves toward $\tau^i(1)$; waiting at a particular $\tau^i(s)$ for some $s^i \in (0,1)$ is also allowed. Each τ^i, is assumed to be a solution to the basic motion planning problem for $\mathcal{A}_i$ (with the other robots removed).

We perform a discrete-time analysis of this problem (similar to that of [8]). The tradeoff is that general completeness is sacrificed, and replaced by *resolution completeness*, which is typically applied to approximate decomposition methods. Thus, $\mathcal{S}$ is represented by a finite number of locations, which correspond to possible positions along the paths at time $k\Delta t$ for some k. For each robot, say $\mathcal{A}_1$, we partition the interval $\mathcal{S}^1 = [0,1]$ into values that are indexed by $i^1 \in \{0, 1, \ldots, i^1_{max}\}$, in which i^1_{max} is given by $\lfloor length(\tau^1)/\|v^1\|\Delta t \rfloor$. Each indexed value yields $\tau^1(i^1\|v^1\|\Delta t/length(\tau^1))$. We denote the discrete-time version of the path coordination space as $\tilde{\mathcal{S}} = \tilde{\mathcal{S}}^1 \times \tilde{\mathcal{S}}^2 \times \cdots \times \tilde{\mathcal{S}}^N$. This yields a restricted space of strategies $\tilde{\Gamma} \subseteq \Gamma$. We consider $\tilde{\mathcal{S}}_{coll}$ and $\tilde{\mathcal{S}}_{valid}$, however, as continuous subsets of $\mathcal{S}$. These can be considered as approximate, cellular representations of $\mathcal{S}_{coll}$ and $\mathcal{S}_{valid}$, respectively (in which cell boundaries are determined by elements in $\tilde{\mathcal{S}}$).

During the time interval $[(k-1)\Delta t, k\Delta t]$ each robot can decide to either remain motionless, or move a distance $\|v^i\|\Delta t$ along the path. The choice taken by a robot, $\mathcal{A}_i$, is referred to as an *action*, which is denoted at stage k as u^i_k. The set of actions for the robots at a given stage is denoted by $u_k = \{u^1_k, \ldots, u^N_k\}$. The choices for u^i_k can be represented as 0, for no motion, and 1 to move forward. We can specialize (4) to obtain the next state from $\tau^i(s^i_k)$, with action u^i_k:

$$f^i(\tau^i(s^i_k), u^i_k) = \left\{ \begin{array}{ll} \tau^i(s^i_k) & \text{if } u^i_k = 0 \\ \tau^i(s^i_k + \|v^i\|\Delta t/length(\tau^i)) & \text{if } u^i_k = 1 \end{array} \right. . \quad (8)$$

We can approximate (5), in discrete time as

$$L^i(\gamma) = \sum_{k=1}^{K} \left\{ l^i_k(x^i_k, u^i_k) + \sum_{j \neq i} c^{ij}_k(x(\cdot)) \right\} + q^i(x^i_{K+1}), \quad (9)$$

in which

$$l^i_k(x^i_k, u^i_k) = \int_{(k-1)\Delta t}^{k\Delta t} g^i(t, x^i(t), u^i(t))dt \quad (10)$$

and

$$c^{ij}_k(x(\cdot)) = \left\{ \begin{array}{ll} 0 & \text{if } x(t) \notin S^{ij}_{coll} \quad \forall t \in [(k-1)\Delta t, k\Delta t] \\ \infty & \text{otherwise} \end{array} \right. . \quad (11)$$

The l^i_k and q^i terms of (9) comprise the standard terms that appear in a discrete-time dynamic optimization context [1]. The middle term, c^{ij}_k represents the interaction between the robots, by penalizing collision. As will be seen shortly, l^i_k will typically be considered as a constant, which for instance, measures time.

We next present an algorithm that determines all of the minimal quotient strategies in $\tilde{\Gamma}/\sim$. We apply the principle of optimality (see [4] for background) to our partially-ordered space of strategies (i.e., portions of minimal strategies are minimal). Both $\tilde{\mathcal{S}}_{coll}$ and $\tilde{\mathcal{S}}$ are represented with N-dimensional arrays. A strategy $\gamma \in \tilde{\Gamma}$ must insure that the robots do not collide during the transitions from x_k to x_{k+1}. (i.e., $x(t)$ does not produce a collision $\forall t \in [(k-1)\Delta t, k\Delta t]$). In practice, this computation depends on the type of curve τ^i, the geometry of $\mathcal{A}_i$, and the type of transformation that is performed to obtain $\mathcal{A}_i(x^i)$.

We construct a data structure that maintains the complete set of minimal quotient strategies from each discretized value, $\tilde{s} \in \tilde{\mathcal{S}}$. Each position $\tilde{s} = (s^1, s^2, \ldots, s^N)$ in the coordination space $\tilde{\mathcal{S}}$ will contain a list of minimal strategies $M(\tilde{s})$, which reach $(1, 1, \ldots, 1)$ from $\tilde{s}$. In $M(\tilde{s})$, we have only one representative strategy for each class in $\tilde{\Gamma}/\sim$. Each element $m \in M(\tilde{s})$ is of the form:

$$m = \langle \, u_k \, , \, [L^{1*} \ L^{2*} \ \cdots \ L^{N*}] \, , \, j \, \rangle. \quad (12)$$

Above, u_k denotes the vector of actions that are to be taken by the robots, in the first step of the strategy represented by m. Each L^{i*} represents the loss that the robot $\mathcal{A}_i$ receives, under the implementation of the minimal strategy that m represents. Using (8), the actions u_k will bring the system to some $\tilde{s}'$. At this state location, there will be a set of strategies represented, $M(\tilde{s}')$, and j above indicates which element in $M(\tilde{s}')$ will continue the strategy.

For a given state, $\tilde{s}$, it will be useful to represent the set of all states that can be reached by trying the various combinations of robot actions that do not yield a collision (one can easily check the array representation of $\tilde{\mathcal{S}}$). We define the *neighborhood* of the state $\tilde{s}$, as the set of immediately reachable states:

$$\mathcal{N}(\tilde{s}_k) = \{\tilde{s}' = f(\tilde{s}, u_k) \mid u_k \in U \text{ and } f(\tilde{s}, u_k) \in \tilde{\mathcal{S}}_{valid}\}, \quad (13)$$

in which f_k represents the next state that is obtained for the vector of robot actions, u_k, and U denotes the space of possible action vectors.

Consider the algorithm in Figure 2. Only a single iteration is required over the coordination space. In Step 1, all states are initially empty, expect for the goal state. Lines 5-8 are iterated over the entire coordination space, starting at the goal state, and terminating at the initial state. At each element, $\tilde{s}$, the minimal strategies are determined by extending the minimal strategies at each neighborhood element.

```
1   Let $M(\tilde{s}_{goal}) = \{\langle \emptyset, [0,0,\ldots,0], \emptyset \rangle\}$, and all other $M(\tilde{s})$ be $\emptyset$
2   For each $i^1$ from $i^1_{max}$ down to 0 do
3     For each $i^2$ from $i^2_{max}$ down to 0 do
        ⋱
4         For each $i^N$ from $i^N_{max}$ down to 0 do
5           Let $\tilde{s} = (i^1, i^2, \ldots, i^N)$
6           Let $M_u$ be a set of strategies that is the union of $M(\tilde{s}')$
              for each $\tilde{s}' \in \mathcal{N}(\tilde{s})$
7           Construct a set $M'_u$ be extending the strategies in $M_u$
8           Let $M(\tilde{s})$ consist of all unique-loss minimal elements of $M'_u$
9   Return $M(\tilde{s}_{init})$
```

Figure 2. A fixed-path coordination algorithm.

Consider the extension of some $m \in M(\tilde{s}')$ in which $\tilde{s}' \in \mathcal{N}(\tilde{s})$. Let u_k be the action such that $\tilde{s}' = f(\tilde{s}, u_k)$. Suppose that m is the i^{th} element in $M(\tilde{s}')$. When $\tilde{s}^{i'} \neq \tilde{s}^i_{goal}$, each loss for the extended strategy is $L^i_k = L^{i'} + l^i_k(\tilde{s}^i, u^i_k)$, otherwise, $L^i_k = 0$. Suppose that m is the j^{th} element in $M(\tilde{s}')$. The third element of m (recall (12)) represents an index, j, which selects a strategy in $M(\tilde{s}')$.

We now discuss how to execute a strategy that is represented as $m \in M(\tilde{s})$. If the action u_k is implemented, then a new state $\tilde{s}'$ will be obtained. The index parameter, j, is used to select the j^{th} element of $M(\tilde{s}')$, which represents the continuation of the minimal strategy. From the j^{th} element of $M(\tilde{s}')$, another action is executed, and a coordination state $\tilde{s}''$ is obtained. This iteration continues until the goal state $(1,1,\ldots,1)$ is reached.

In Figure 3.a we show an example in which there are three robots. The initial positions are indicated in Figure 3.a: $\mathcal{A}_1$ is black, $\mathcal{A}_2$ is white, and $\mathcal{A}_3$ is gray. Figure 3.b shows the computed representation of $\tilde{S}$. The axes show distances along the paths. The cylindrical structure in $\tilde{S}_{coll}$ can be clearly observed in this example. The two vertical columns correspond to the two collisions that can occur between $\mathcal{A}_1$ and $\mathcal{A}_2$. Each of the two horizontal columns represents collisions of $\mathcal{A}_3$ with $\mathcal{A}_1$ or $\mathcal{A}_2$. There are two minimal quotient strategies for this problem, for which representative strategies are depicted as paths in the coordination space.

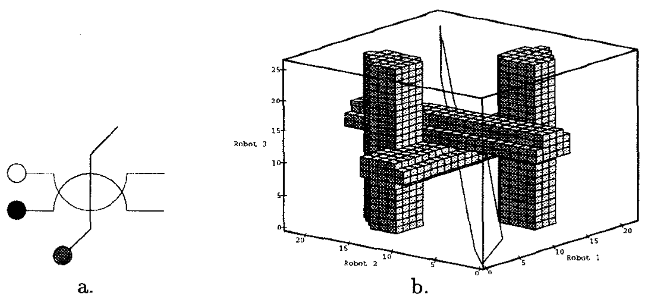

Figure 3. A coordination space for a three-robot problem.

4 Motion Planning Along Roadmaps

In this section we present a method that determines minimal strategies for the case in which the robots are restricted to independent roadmaps. This allows many more strategies to be considered than for fixed path coordination, while only causing a modest increase in computation. Many of the general concepts are similar to those from the last section; however, the complicated topological structure of a cartesian product of roadmaps makes this problem more complex. At the end of the section, several computed minimal strategies are shown.

We consider a *roadmap* for $\mathcal{A}_i$ to be a collection of curves, $\mathcal{T}^i$, such that for each $\tau^i_j \in \mathcal{T}^i$, $\tau^i_j : [0,1] \rightarrow \mathcal{C}^i_{valid}$. We assume (without loss of generality) that each parametrization is of constant speed. The endpoints of some paths coincide in $\mathcal{C}^i_{valid} = X^i$, to form a network.

We let $\mathcal{R}^i$ denote a set that represents the union of transformed domains of the paths in $\mathcal{T}^i$. Using the $\mathcal{R}^i$'s, we can describe a *roadmap coordination space*, $\mathcal{R} = \mathcal{R}^1 \times \mathcal{R}^2 \times \cdots \times \mathcal{R}^N$. We can specify a position $r \in \mathcal{R}$ by $r = (\pi^1, \pi^2, \ldots, \pi^N; s^1, s^2, \ldots, s^N)$. Each π^i specifies the path in $\mathcal{R}^i$ that $\mathcal{A}_i$ has chosen, while each s^i specifies the position of the robot along that path. A problem is specified by providing an initial configuration, $r^i_{init} \in \mathcal{R}^i$, and a goal configuration $r^i_{goal} \in \mathcal{R}^i$ for each robot, $\mathcal{A}_i$.

We construct the discrete representations, $\tilde{\mathcal{R}}_{coll}$ and $\tilde{\mathcal{R}}$, which are similar to $\tilde{S}_{coll}$ and $\tilde{S}$. We build one array for each combination of path choices for the robots, each of which can be constructed in the same manner as for $\tilde{S}_{coll}$ and $\tilde{S}$. This representation can be intuitively be thought of as a network of coordination spaces that exists in $\Re^N$.

There are two primary differences between the roadmap coordination problem and the fixed path coordination problem. The first difference is that robots on $\mathcal{R}$ are allowed to move in either direction along a path, usually resulting in $3^N - 1$ choices for u_k as opposed to $2^N - 1$ (there even more choices when one or more robots moves into a junction, because a new path must be selected). This leads to more complicated strategies; for instance, if $l^i_k(x^i_k, u^i_k) = \Delta t$, then there will in general exist minimal strategies in Γ that cause one or more robots to oscillate on a path. The second major difference is the complicated topology of $\mathcal{R}$, which leads to a complicated neighborhood structure.

We now describe the algorithm in Figure 4. A set of roadmap coordination states, termed a *wavefront*, $\mathcal{W}_i$, is maintained in each iteration. During an iteration, the complete set of minimal strategies is determined for each element of $\mathcal{W}_i$. The initial wavefront, $\mathcal{W}_0$, contains only the goal state. Each new wavefront $\mathcal{W}_i$ is defined as the set of all states that: 1) can be reached in one stage from an element in $\mathcal{W}_i$, and 2) are not included in any of $\mathcal{W}_{i-1}, \ldots, \mathcal{W}_0$. The algorithm terminates when all states have been considered. This algorithm could be viewed as a multiple-objective extension of the wavefront algorithm that is used in [3].

We present some computed examples that were obtained with the algorithm in Figure 4. Figure 5 presents one minimal strategy in a roadmap coordination problem that involves three robots in $\Re^3$, with different roadmaps for each robot.

Figure 6 presents an example in which there are two robots in the plane that move along independent roadmaps. The configuration spaces of the individual robots is three dimensional in this case because robots can rotate while moving along the roadmap. There are five minimal quotient strategies for this problem, and the two that are shown do not require either robot to wait. Quite distinct routes, however,

```
Initialize $\tilde{\mathcal{R}}$
Let $\mathcal{W}_0 = \{\tilde{r}_{init}\}$
$i = 0$
Until $\mathcal{W}_i = \emptyset$ do
  For each $\tilde{r} \in \mathcal{W}_i$ do
    Let $M_u$ be a set of strategies that is the union of $M(\tilde{r}')$
      for each $\tilde{r}' \in \mathcal{N}(\tilde{r})$
    Construct a set $M_u'$ by extending the strategies in $M_u$
    Let $M(\tilde{r})$ consist of all unique-loss minimal elements of $M_u'$
    Let $i = i + 1$
  Let $\mathcal{W}_i$ be set of all neighbors of $\mathcal{W}_{i-1}$ that have not yet
      been processed
Return $M(\tilde{r}_{init})$
```

Figure 4. An algorithm for independent roadmaps.

are taken by the robots in the different strategies.

Figure 7 shows an "H"-shaped roadmap coordination solution with rotating robots (from left to right). This problem is perhaps one of the most complex in terms of solution alternatives; one minimal quotient strategy out of sixteen is represented in the figure.

5 Centralized Motion Planning

In this section, we present an algorithm that determines one minimal strategy on the unconstrained state space, $X = \mathcal{C}^1_{valid} \times \mathcal{C}^2_{valid} \times \cdots \times \mathcal{C}^N_{valid}$. We present some computed examples for the case in which there are two translating robots in $\Re^2$.

A vector β is chosen such that a linear scalarizing function, H, is defined. We allow each robot goal to be a subset, $X^i_G \subset X^i$. We approximate (4) by discrete-time state transition equations, $x^i_{k+1} = f^i_k(x^i_k, u^i_k)$. We define the action space for robot $\mathcal{A}^i$ as $U^i = [0, 2\pi) \cup \{\emptyset\}$. If $u^i_k \in [0, 2\pi)$, then $\mathcal{A}^i$ attempts to move a distance $\|v^i\|\Delta t$ toward a direction in $\mathcal{C}^i$, in which $\|v^i\|$ denotes some fixed speed for $\mathcal{A}^i$. If $u^i_k = \emptyset$, then the robot remains motionless.

Suppose that at some stage k, the optimal strategy is known for each stage $i \in \{k, \ldots, K\}$. The loss obtained by starting from stage k, and implementing the portion of the optimal strategy, $\{\gamma^*_k, \ldots, \gamma^*_K\}$, can be represented as

$$L^{i*}_k(x_k) = \sum_{k'=k}^{K} \left\{ l^i_{k'}(x^i_{k'}, u^i_{k'}) + \sum_{j \neq i} c^{ij}_{k'}(x(\cdot)) \right\} + q^i(x^i_{K+1}). \tag{14}$$

The function $L^{*i}_k(x_k)$ is sometimes referred to as the *cost-to-go* function in dynamic optimization literature [4]. For this context, we modify the definition of $q^i(x^i_{K+1})$, by replacing $x^i(T) = x^i_{goal}$ with $x^i(T) \in X^i_G$.

We can convert the cost-to-go functions into a scalar function by applying $H(\gamma, \beta)$ to obtain

$$H^*_k = \sum_{i=1}^{N} \beta_i L^{i*}_k(x_k). \tag{15}$$

Above, H^*_k represents a single cost-to-go function, which implicitly assumes that β is given.

The principle of optimality [1] implies that $H^*_k(x_k)$ can be obtained from $H^*_{k+1}(\cdot)$ by selecting an optimal value for u_k. The following recurrence represents the principle of optimality for our context:

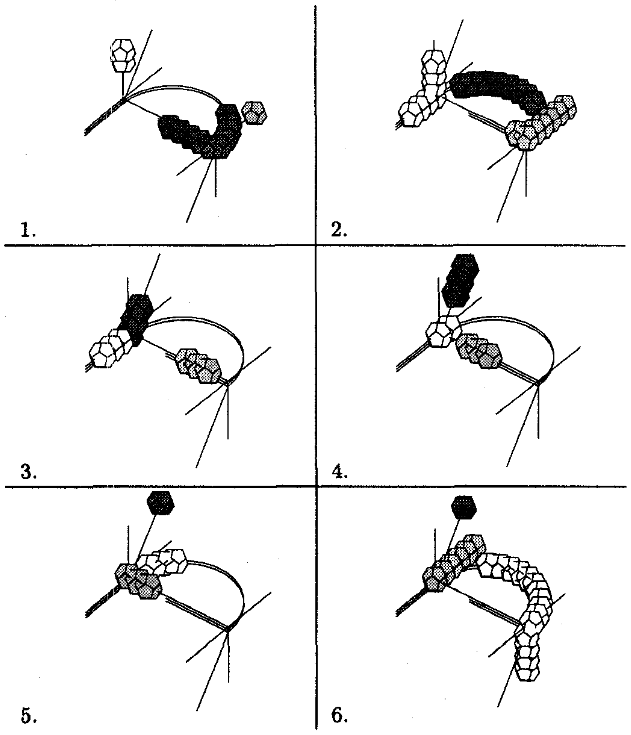
1.
2.
3.
4.
5.
6.

Figure 5. A representative of one of four minimal quotient strategies.

$$H^*_k(x_k) = \min_{u_k \in U}$$

$$\left\{ \sum_{i=1}^{N} \beta_i l^i_k(x_k, u_k) + \sum_{i=1}^{N} \sum_{j \neq i} \beta_i c^{ij}_k(x^i(\cdot)) + H^*_{k+1}(x_{k+1}) \right\}. \tag{16}$$

For each choice of u_k, x_{k+1} is obtained by applying f^i_k for each $i \in \{1, \ldots, N\}$. The boundary condition for this recurrence is given by applying the scalarizing function to the q^i's. The cost-to-go function $H^*_1(x_1)$ shares similarities with the concept of a global navigation function in motion planning [9].

We determine optimal strategies numerically, by successively building approximate representations of H^*_k for each dynamic programming iteration. We obtain the value for $H^*_k(x_k)$ by computing the right side of (16) for various values of u_k, including $u_k = \emptyset$. The value for $H^*_k(x_k)$ is obtained by linear interpolation. For a stationary strategy, the optimal action u_k, does not depend on the stage index, k. Hence, we perform dynamic programming iterations over the state space until there are no locations at which H^*_{k+1} is less than H^*_k. To execute a strategy, the robots use the final cost-to-go representation, which we call H^*_1. The optimal u_k can be obtained from any real-valued location $x \in X$ though the use of (16), linear interpolation, and the approximate representation of H^*_1.

We present two computed examples that were obtained with the algorithm described in this section. Both examples involve motion planning for two robots, which are allowed to independently translate in $\Re^2$ (without restriction to a

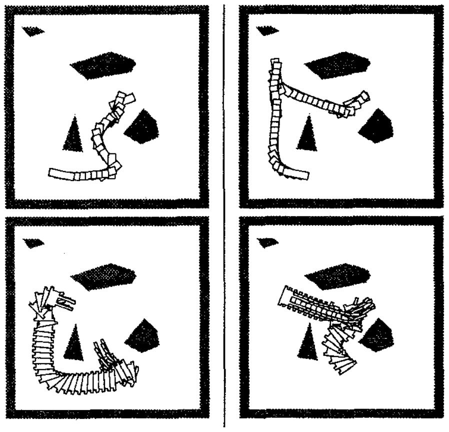
Figure 6. Two of five solutions.

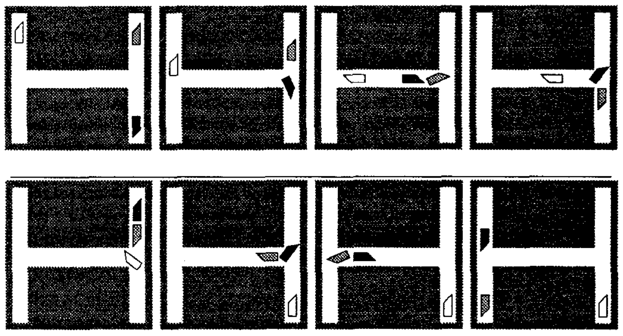
Figure 7. One of sixteen solutions.

path or roadmap). For the problem in the left of Figure 8, we set $\beta_1 = \beta_2 = \frac{1}{2}$. In the solution, neither robot is required to wait; they move around each other. Figure 8 shows another computed example. The first solution is found when $\beta_1 = \beta_2 = \frac{1}{2}$, which is equivalent to minimizing the total aggregate time. The second solution corresponds to $\beta_1 = \frac{19}{20}$ and $\beta_2 = \frac{1}{20}$. For this case $\mathcal{A}_1$ (which is initially the rightmost robot) receives a greater priority, and is allowed to execute its time-optimal solution, while $\mathcal{A}_2$ is forced to wait.

6 Conclusions

We have presented a general method for multiple-robot motion planning that is centered on a concept of optimality with respect to independent objectives. Strategies are determined that simultaneously optimize an independent performance criterion for each robot. In addition, a general spectrum has been defined between decoupled and centralized planning, in which we introduce optimal coordination along independent roadmaps.

Acknowledgements

We gratefully acknowledge Narendra Ahuja, Tamer Başar, Mike Erdmann, Dan Koditschek, Jean Ponce, Rajeev Sharma, Steve Sullivan, and Mark Spong, for their helpful comments and suggestions. This work was sponsored by NSF under grant #IRI-9110270, a Beckman Institute research assistantship, and a Mavis Fellowship.

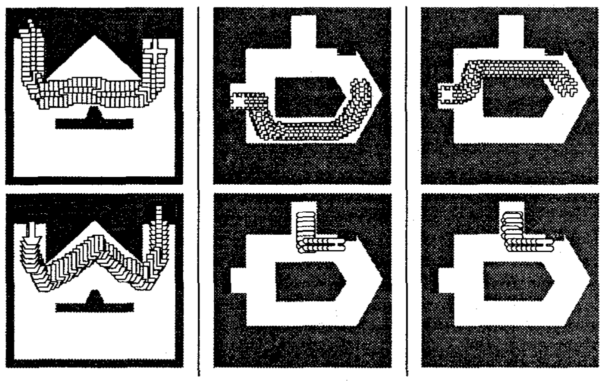
Figure 8. A minimal quotient strategy is depicted for one problem, and two alternative solutions are presented for another.

References

[1] T. Başar and G. J. Olsder. *Dynamic Noncooperative Game Theory.* Academic Press, London, 1982.

[2] J. Barraquand, B. Langlois, and J. C. Latombe. Numerical potential field techniques for robot path planning. *IEEE Trans. Syst., Man, Cybern.*, 22(2):224–241, 1992.

[3] J. Barraquand and J.-C. Latombe. Robot motion planning: A distributed representation approach. *Int. J. Robot. Res.*, 10(6):628–649, December 1991.

[4] D. P. Bertsekas. *Dynamic Programming: Deterministic and Stochastic Models.* Prentice-Hall, Englewood Cliffs, NJ, 1987.

[5] Z. Bien and J. Lee. A minimum-time trajectory planning method for two robots. *IEEE Trans. Robot. & Autom.*, 8(3):414–418, June 1992.

[6] M. Erdmann and T. Lozano-Perez. On multiple moving objects. In *IEEE Int. Conf. Robot. & Autom.*, pages 1419–1424, 1986.

[7] S. M. LaValle. *A Game-Theoretic Framework for Robot Motion Planning.* PhD thesis, University of Illinois, Urbana, IL, July 1991.

[8] P. A. O'Donnell and T. Lozano-Pérez. Deadlock-free and collision-free coordination of two robot manipulators. In *IEEE Int. Conf. Robot. & Autom.*, pages 484–489, 1989.

[9] E. Rimon and D. E. Koditschek. Exact robot navigation using artificial potential fields. *IEEE Trans. Robot. & Autom.*, 8(5):501–518, October 1992.

[10] Y. Sawaragi, H. Nakayama, and T. Tanino. *Theory of Multiobjective Optimization.* Academic Press, New York, NY, 1985.

[11] K. G. Shin and Q. Zheng. Minimum-time collision-free trajectory planning for dual-robot systems. *IEEE Trans. Robot. & Autom.*, 8(5):641–644, October 1992.

[12] F.-Y. Wang and P. J. A. Lever. A cell mapping method for general optimum trajectory planning of multiple robotic arms. *Robots and Autonomous Systems*, 12:15–27, 1994.

[13] S. Zionts. Multiple criteria mathematical programming: An overview and several approaches. In P. Serafini, editor, *Mathematics of Multi-Objective Optimization*, pages 227–273. Springer-Verlag, Berlin, 1985.