Optimal Motion Planning for Multiple Robots Having Independent Goals

Steven M. LaValle
Dept. of Computer Science
Iowa State University
Ames, IA 50011
lavalle@cs.iastate.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 are computed that simultaneously optimize an independent performance measure for each robot; ii) A general spectrum is defined between decoupled and centralized planning, in which we introduce coordination along independent roadmaps.

By considering independent performance measures, we introduce a form of optimality that is consistent with concepts from multi-objective optimization and game theory literature. Previous multiple-robot motion planning approaches that consider optimality combine individual performance measures into a scalar 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. Computed examples are presented for all three problem classes that illustrate the concepts and algorithms.

1 Introduction

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 a real-valued performance measure (or loss functional) to be optimized.

This final point differs from previous approaches to multiple-robot motion planning. Typically, if optimality is considered, individual performance measures for the robots are combined into a single scalar criterion. For example, in [6, 21] the criterion is to minimize the time taken by the last robot to reach the goal. In [24], the performance measures are added to yield a scalar criterion. When individial performance measures are combined, certain information about potential solutions and alternatives is lost (for a general discussion, see [19]). For example, the amount of sacrifice that each robot makes to avoid other robots is not usually taken into account. It might be that one robot's goal is nearby, while the other robot has a distant goal. Combining the performance measures might produce a plan that is good for the robot that has the distant goal; however, the performance of the other robot would be hardly considered.

Given a vector of independent performance measures, we show that there exists a natural partial ordering on the space of motion plans, yielding a search for the set of motion plans that are *minimal* with respect to the ordering. Our approach can be considered as filtering out all of the motion plans that are not worth considering, and presenting the user with a small set of the best alternatives. Within this framework additional criteria, such as priority or the amount of sacrifice one robot makes, can be applied to automatically select a particular motion plan. If the same tasks are repeated and priorities change, then only needs to select an alternative minimal plan, as opposed to re-exploring the entire space of motion strategies. We also show that the minimal strategies are consistent with certain optimality concepts from multiobjective optimization (e.g., [19]) and dynamic game theory (e.g., [2]) literature.

Previous approaches to multiple-robot motion planning are often categorized as *centralized* or *decoupled*. A centralized approach typically constructs a path in a composite configuration space, which is formed by forming the Cartesian product of the configuration spaces of the individual robots (e.g., [1, 3, 20]). A decoupled approach typically generates paths for each robot independently, and then considers the interactions between the robots (e.g., [7, 10]). In [6, 9, 16] robot paths are independently determined, and a coordination diagram is used to plan a collision-free trajectory along the paths. The suitability of one approach over the other is usually determined by the tradeoff between computational complexity associated with a given problem, and the amount of completeness that is lost.

In addition to introducing multiple-objective optimality to the multiple-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 only 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 travel on independent networks of paths called *roadmaps*. Many approaches exist that construct roadmaps for a single robot (e.g., [8, 17]), which can be used as a preprocessing step in our coordination approach.

Our algorithms are based on applying the dynamic programming principle to generate multiple solutions in a partially-ordered space of motion strategies. The generation of these solutions is significantly more challenging in comparison to the standard case of scalar optimization. Many variations of dynamic programming for scalar optimization have been applied in motion planning (e.g., [11, 15, 22]) and in AI planning (e.g., [4, 5, 23]); however, techniques are presented in this paper to derive multiple solutions for the case of multiple, independent performance measures.

2 Problem Definition and General Concepts

2.1 Basic Definitions

Each robot, $\mathcal{A}_i$, is considered as a rigid object, capable of moving in a workspace that is a bounded subset of $\Re^2$ or $\Re^3$. The position and orientation of the robot in the workspace are specified parametrically, by a point in an n-dimensional *configuration space*, $\mathcal{C}^i$. There are static obstacles in the workspace (compact subsets of $\Re^2$ or $\Re^3$) that prohibit certain configurations of the robot, $\mathcal{A}_i$. The closure of the subset of $\mathcal{C}^i$ that corresponds to configurations in which $\mathcal{A}_i$ does not intersect any obstacles is referred to as the *valid configuration space*, $\mathcal{C}^i_{valid}$ [13].

We define a *state space*, X, that simultaneously represents the configurations of all of the robots. A natural choice for the state space is

$$X = \mathcal{C}^1_{valid} \times \mathcal{C}^2_{valid} \times \cdots \times \mathcal{C}^N_{valid}, \tag{1}$$

in which $\times$ denotes the Cartesian product. In this paper, we also consider two additional definitions of the state space that are more restrictive. In Section 3 we will consider motions of the robots that are restricted to fixed paths, and in Section 4 we will consider a more general case in which the robots are constrained to move along independent roadmaps.

The concepts introduced in the remainder of this section apply to any of the above state space definitions. For this reason we generally refer to the state space as

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

and use the notation $\mathcal{A}_i(x^i)$ to refer to the transformed robot, $\mathcal{A}_i$, at configuration x^i.

In multiple robot motion planning problems, we are not only concerned about collision with obstacles, but also about collisions that occur between robots. 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 3)

$$X_{coll}^{ij} = \{x \in X \mid \mathcal{A}_i^\circ(x^i) \cap \mathcal{A}_j^\circ(x^j) \neq \emptyset\}, \tag{3}$$

which represents the set of states in which the two robots collide. The reason for using the interior of $\mathcal{A}_i$ is to allow the robots to "touch". The *collision subset*, $X_{coll} \subset X$, is represented as the open set,

$$X_{coll} = \bigcup_{i \neq j} X_{coll}^{ij}. \tag{4}$$

Hence, a state is in the collision subset if the interior of two or more robots intersect. We define X_{valid} as the closed set, $X - X_{coll}$. Note the cylindrical structure of X_{coll} (depicted in Figure 3), which is exploited by our algorithms when building a representation of the state space, allowing the number of collision detections to grow quadratically with N, as opposed to exponentially.

The task is to bring each robot from some initial state $x_{init}^i \in X^i$ to some goal state $x_{goal}^i \in X^i$ while avoiding collisions with obstacles or other robots. We consider a *state trajectory* as a continuous mapping $x : [0, T] \rightarrow X$. A trajectory for an individual robot is represented as $x^i : [0, T] \rightarrow X^i$. The motion of an individual robot, $\mathcal{A}_i$, is specified through the *state transition equation*,

$$\dot{x}^i(t) = f^i(x^i(t), u^i(t)) \;\; \forall i, \tag{5}$$

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

Since we focus on the geometric aspects of a motion planning problem, we will compute trajectories that apparently allow a robot to switch instantaneously between a fixed speed $\|v^i\|$ and halting. This represents a typical assumption in multiple-robot motion planning [10, 12, 16]. In a sense, the results we ultimately obtain will involve both path and scheduling information. For most mechanical systems, the dynamics must be taken into account at some level, and in this paper we choose to decouple the general pick-and-place problem into two modules: 1) motion planning/trajectory generation, and 2) a tracking controller. This is a widely-utilized assumption that forms the basis of motion planning research (see [13] and references therein). We expect that in many applications, especially mobile robotics, optimal solutions generated with the first module will be suitable for an integrated system. However, in general, we concede that the resulting solutions might not be feasible for many applications in which the dynamic constraints prohibit tracking of our designed trajectories.

To evaluate the performance of each robot, $\mathcal{A}_i$, we define a *loss functional* of the form

$$L^i(x_{init}, x_{goal}, u^1, \ldots, u^N) = \int_0^T g^i(t, x^i(t), u^i(t))dt + \sum_{j \neq i} c^{ij}(x(\cdot)) + q^i(x^i(T)), \tag{6}$$

which maps to the extended reals, and $c^{ij}(x(\cdot)) = 0$ if $x(t) \in X_{valid}$ for all t, and ∞ 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

$$g^i(t, x^i(t), u^i(t)) = 0 \quad \text{if } x^i(t) = x^i_{goal}. \tag{7}$$

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 (6), $c^{ij}(x(\cdot))$ penalizes collisions between the robots. The function $q^i(x^i(T))$ in (6) represents the goal in terms of performance. If $\mathcal{A}_i$ fails to achieve its goal, x^i_{goal}, then it receives infinite loss.

2.2 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}. The goal of each robot is to choose some control function, u^i that achieves the goal x^i_{goal} while trying to minimize the loss functional (6). We will use the notation γ^i to refer to a *robot strategy* for $\mathcal{A}_i$, which represents a possible choice of u^i 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.

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}.

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). 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.

Consider a strategy, γ, which produces $L^1(\gamma) = 1$ and $L^2(\gamma) = 2$, and another strategy, γ', which produces $L^1(\gamma') = 2$ and $L^2(\gamma') = 1$. From a global perspective, it is not clear which strategy would be preferable. Robot $\mathcal{A}_1$ would prefer γ, while $\mathcal{A}_2$ would prefer γ'. Both robots would, however, prefer either strategy to a third alternative that produced $L^1(\gamma'') = 5$ and $L^2(\gamma'') = 5$. These comparisons suggest that there exists a natural partial ordering on the space of strategies. Our interest is in finding the set of strategies that are minimal with respect to this partial ordering; these comprise all of the useful strategies, since any other strategies would not be preferred by any of the robots.

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 γ is better than γ' if $[\gamma]_L \preceq [\gamma']_L$. A quotient strategy, $[\gamma^*]_L$, is *minimal* if for all $[\gamma]_L \neq [\gamma^*]_L$ such that $[\gamma]_L$ and $[\gamma^*]_L$ are not incomparable, then $[\gamma^*]_L \preceq [\gamma]_L$.

2.3 Relationships to Established Forms of Optimality

In this section we briefly state how the minimal strategies relate to optimality concepts from multiobjective optimization and dynamic game theory; a more thorough discussion appears in [14]. The minimal quotient strategies are equivalent to the *nondominated* strategies used in multiobjective optimization and *Pareto optimal* strategies used in cooperative game theory. Furthermore, we have shown [14] that under the general loss functional (6), 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 $\gamma^i \in \Gamma^i$:

$$L^i(\gamma^{1*}, \ldots, \gamma^{i*}, \ldots \gamma^{N*}) \leq L^i(\gamma^{1*}, \ldots, \gamma^{i}, \ldots \gamma^{N*}). \tag{8}$$

We can also consider the relationship between our minimal strategies, and scalar optimization. In multiobjective optimization literature, this is referred to as *scalarization* [19], in which a mapping that projects the loss vector to a scalar, while guaranteeing that optimizing the scalar loss produces a nondominated strategy. This function is used in Section 5, in an algorithm that determines minimal strategies. 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 a weighted-average of losses among the robots:

$$H(\gamma, \beta) = \sum_{i=1}^{N} \beta_i L^i(\gamma). \tag{9}$$

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 [14].

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 some 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}. We have also shown through homotopy that few minimal quotient strategies will typically exist, and present an algorithm that determines the minimal quotient strategies.

3.1 Concepts and Definitions

We assume that each robot, $\mathcal{A}_i$, is given a path, τ^i, which is a continuous mapping $[0, 1] \rightarrow \mathcal{C}^i_{valid}$. 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. It is assumed that the robots do not collide with static obstacles, implying that each given path, τ^i, is 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, and partition $[0, T]$ into *stages*, denoted by $k \in \{1, \ldots, K\}$. Stage k refers to time $(k - 1)\Delta t$. The development of analytical, continuous-time solutions would require detailed analysis for specific models and geometric representations; however, with discrete time, we can readily compute solutions to a variety of motion planning problems. The discrete-time representation induces a discretization of the state space, which is typically obtained in motion planning research (e.g., [16]). The tradeoff is that general completeness is sacrificed, and replaced by *resolution completeness*, which is typically applied to approximate decomposition methods [13]. This implies that our method will find solutions that exist at a certain resolution, and this resolution can be arbitrarily improved. We assume that we can send an action (or motion command) to each robot every Δt seconds.

Discretized time allows $\mathcal{S}$ to be 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 approximation of the path coordination space as $\tilde{\mathcal{S}}$. 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 (5) to obtain the next state from $\tau^i(s^i_k)$, with action u^i_k:

$$f^i(\tau^i(s^i_k), u^i_k) = \begin{cases} \tau^i(s^i_k) & \text{if } u^i_k = 0 \\ \\ \tau^i(s^i_k + \|v^i\|\Delta t/length(\tau^i)) & \text{if } u^i_k = 1 \end{cases} . \tag{10}$$

We can approximate (6), in discrete time as

$$L^i(\gamma) = \sum_{k=1}^{K} \left\{ l^i_k(x^i_k, u^i_k) + \sum_{j \neq i} c^{ij}_k(x(\cdot)) \right\} + q^i(x^i_{K+1}), \tag{11}$$

in which

$$l^i_k(x^i_k, u^i_k) = \int_{(k-1)\Delta t}^{k\Delta t} g^i(t, x^i(t), u^i(t))dt \tag{12}$$

and

$$c^{ij}_k(x(\cdot)) = \begin{cases} 0 & \text{if } x(t) \notin S^{ij}_{coll} \quad \forall t \in [(k-1)\Delta t, k\Delta t] \\ \infty & \text{otherwise} \end{cases} . \tag{13}$$

The l^i_k and q^i terms of (11) comprise the standard terms that appear in a discrete-time dynamic optimization context [2]. 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.

Before discussing the algorithm in Section 3.2, we will provide a proposition that characterizes the quantity of minimal quotient strategies that can exist in $\tilde{\Gamma}/\sim$, for the fixed-path coordination problem. It might appear that there could be numerous minimal quotient strategies, even for only two robots. For instance, suppose there were strategies that produced losses $L^1 = i$ and $L^2 = 10000 - i$ for each $i \in \{1, \ldots, 10000\}$. No pair of these strategies are comparable, and hence they could all be minimal. In multiobjective optimization the existence numerous or even an infinite number of solutions often causes difficulty [25]. We show that at least for the case in which time-optimality is of interest (i.e., $l_k^i(x_k^i, u_k^i) = \Delta t$ for all i, k), there are very few minimal quotient strategies because each must be obtained from a distinct path class in $\tilde{\mathcal{S}}_{valid}$.

A given strategy $\gamma \in \tilde{\Gamma}$ yields a trajectory $\alpha_\gamma : [0, T] \to \mathcal{S}$ through the coordination space. A different strategy, $\gamma' \in \tilde{\Gamma}$ yields a trajectory $\alpha_{\gamma'}$. The two paths α_γ and α'_γ are *homotopic* in $\tilde{\mathcal{S}}_{valid}$ (with endpoints fixed) if there exists a continuous map $h : [0, T] \times [0, 1] \to \tilde{\mathcal{S}}_{valid}$ with $h(t, 0) = \alpha_\gamma(t)$ and $h(t, 1) = \alpha_{\gamma'}(t)$ for all $t \in [0, T]$, and $h(0, s) = h(0, 0)$ and $h(1, s) = h(1, 0)$ for all $s \in [0, 1]$. This homotopy determines an equivalence relation on the state trajectories, and hence on the space of strategies, $\tilde{\Gamma}$. Note that since α_γ is monotone, the path classes defined here do *not* represent the fundamental group from homotopy theory; there are far fewer path classes in this context.

Using these path classes we have the following proposition:

Proposition 1 *If $l_k^i(x_k^i, u_k^i) = \Delta t$ for all $i \in \{1, \ldots, N\}$ and $k \in \{1, \ldots, K\}$, then there exists at most one minimal quotient strategy per path class in $\tilde{\mathcal{S}}_{valid}$.*

The proof of this and remaining propositions appear in [14]. Hence the number of solutions is quite restricted, and in practice there are typically only a few minimal quotient strategies.

3.2 Algorithm Presentation

In this section we present an algorithm that determines all of the minimal quotient strategies in $\tilde{\Gamma}/\sim$ by applying the dynamic programming principle to the partially-ordered strategy space. We represent both $\tilde{\mathcal{S}}_{coll}$ and $\tilde{\mathcal{S}}$ as N-dimensional arrays. A strategy $\gamma \in \tilde{\Gamma}$ must ensure 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:

$$m = \langle\ u_k\ ,\ [L^{1*}\ \ L^{2*}\ \ \cdots\ \ L^{N*}]\ ,\ j\ \rangle. \tag{14}$$

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 (10), the actions u_k will bring the system to some $\tilde{s}'$. At this location, there will be a set, $M(\tilde{s}')$, of strategies represented, 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 $\tilde{\mathcal{S}}$). Define $\mathcal{N}(\tilde{s}) \subset \tilde{\mathcal{S}}$ as the *neighborhood* of the state $\tilde{s}$, which corresponds to these immediately reachable states. Formally we have

$$\mathcal{N}(\tilde{s}_k) = \{\tilde{s}' = f(\tilde{s}, u_k) \mid u_k \in U \text{ and } f(\tilde{s}, u_k) \in \tilde{\mathcal{S}}_{valid}\}, \tag{15}$$

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 1. Only a single iteration is required over the coordination space. The algorithm terminated when the minimal quotient strategies have been computed from each state that is connected to the goal. Note that this algorithm does not require one to determine K in advance. In Line 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.

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 $M(\tilde{s}')$. The loss for the extended strategy is given by

$$L^i_k = \begin{cases} 0 & \text{if } \tilde{s}^{i\prime} = 1 \\ L^{i\prime} + l^i_k(\tilde{s}^i, u^i_k) & \text{otherwise} \end{cases}, \tag{16}$$

for each $i \in \{1, \ldots, N\}$. Suppose that m is the j^{th} element in $M(\tilde{s}')$. The third element of m (recall (14)) 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.

3.3 Computed Examples

The algorithms presented in this paper were quickly implemented in Common Lisp on a SPARC 10 workstation with only 84 FL MIPS and 73.7 IN MIPS. No consideration was given to reducing computation time; however, the computation times are given for comparisons between examples.

In Figure 4.a we show an example in which there are three robots. The initial positions are indicated in Figure 4.a: $\mathcal{A}_1$ is black, $\mathcal{A}_2$ is white, and $\mathcal{A}_3$ is gray. Figure 4.b shows the computed representation of $\tilde{\mathcal{S}}$. The axes show distances along the paths. The cylindrical structure in $\tilde{\mathcal{S}}_{coll}$ 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 were 3125 collision checks which took 18s, and the solution computation took 9s. There are two minimal quotient strategies for this problem, for which representative strategies are depicted as paths in the coordination space.

Figure 5 shows a three-robot example in which two robots move along "S"-curves, and the third robot moves horizontally. There were 17721 collision checks which took 124s, and the solution computation took 37s. There are four minimal quotient strategies for this problem, which produce losses:

Strategy	**Loss 1**	**Loss 2**	**Loss 3**
γ_1^*	81	75	30
γ_2^*	79	73	82
γ_3^*	83	73	41
γ_4^*	73	80	30

.

Each integer represents the number of stages required to reach x^i_{goal}. In the lower portion of Figure 8, we show four sets of timing diagrams, each of which corresponds to a representative minimal quotient strategy that was computed. Each graph indicates whether a robot is moving or waiting, as a function of time.

4 Motion Planning Along Independent Roadmaps

In this section we present a method that determines minimal strategies for the case in which each robot is constrained to traverse a network of collision-free paths. Many of the general concepts are similar to those from the last section; however, the topological structure of a Cartesian product of roadmaps makes this problem more complex.

4.1 Concepts and Definitions

We consider a *roadmap* for $\mathcal{A}_i$ to be a collection of constant-speed curves, $\mathcal{T}^i$, such that for each $\tau^i_j \in \mathcal{T}^i$, $\tau^i_j : [0,1] \rightarrow \mathcal{C}^i_{valid}$. The endpoints of some paths coincide in $\mathcal{C}^i_{valid}$, to form a network.

Recall that in the previous section we considered robot coordination on the Cartesian product of unit intervals, which represented the domains of the paths. For the roadmap coordination problem, we will coordinate the robots on the domains of the functions in $\mathcal{T}^i$. 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*, $\mathcal{R} = \mathcal{R}^1 \times \mathcal{R}^2 \times \cdots \times \mathcal{R}^N$. A position $r \in \mathcal{R}$ in indicated by specifying both a path and a position along that path, for each of the robots.

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, $\mathcal{A}_i$. An individual roadmap could also be extended to cover a new initial or goal position in a motion planning query [13]. During the time interval $[(k-1)\Delta t, k\Delta t]$ each robot can decide to either remain motionless, or move a distance $\|v^i\|\Delta t$ in either direction along a path. Also, if the robot moves into a roadmap junction, then a new path must be chosen.

4.2 Algorithm Presentation

We consider the case in which $l^i_k(x^i_k, u^i_k) = \Delta t$ for all i, k. We construct the discrete representations, $\tilde{\mathcal{R}}_{coll}$ and $\tilde{\mathcal{R}}$, which are similar to $\tilde{\mathcal{S}}_{coll}$ and $\tilde{\mathcal{S}}$, and build one array for each combination of path choices for the robots, each of which can be constructed in the same manner as for $\tilde{\mathcal{S}}_{coll}$ and $\tilde{\mathcal{S}}$. This representation can be considered as a network of coordination spaces.

There are two primary differences between the roadmap coordination problem and the fixed path coordination problem in terms of the algorithm development. The first difference is that robots on $\mathcal{R}$ are allowed to move in either direction. For fixed paths, we assumed that the robots could only move forward along a path. By allowing the robots to move in either direction, there are usually $3^N - 1$ choices for u_k as opposed to $2^N - 1$ (there are additional choices when one or more robots moves into a junction, because a new path must be selected). The second major difference is the complicated topology of $\mathcal{R}$, as opposed to $\mathcal{S}$ which is a unit cube.

Both of these differences increase the difficulty of defining the neighborhood of a state. For an example of a neighborhood in the roadmap coordination problem in which $N = 2$, consider Figures 6 and 7. For this example, the second robot is approaching a junction, while the first robot is in the middle of a path. The white circles in Figure 6 indicate the positions of the robots at state $\tilde{r}$, and the black circles indicate possible locations of the robots at the next state, $\tilde{r}'$. The representation of this situation in $\tilde{\mathcal{R}}$ is shown in Figure 7. For this problem, there are 11 possible choices for $\tilde{r}'$. For each representation of some $m \in M(\tilde{s})$, in addition to the components in (14), we store an index when necessary that indicates which new paths are chosen by the robots.

The algorithm is described in Figure 2. 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].

4.3 Computed Examples

We present some computed examples that were obtained with the algorithm in Figure 2. There were 1620 collision checks which took 18s, and the solution computation took 17s. Figure 8 shows the two unique-loss minimal strategies side-by-side, for an "H"-shaped roadmap coordination problem in which two robots attempt to reach opposite corners. The black and white discs represent $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively. The black and white triangles indicate the goal configurations. Intuitively, for this problem, one would expect two symmetric possibilities to exist: either $\mathcal{A}_1$ has to wait, or $\mathcal{A}_2$ has to wait. These two situations are precisely what are obtained in the two minimal quotient strategies.

Figures 9 and 10 present one minimal strategy in a roadmap coordination problem that involves three robots in $\Re^3$, with different roadmaps for each robot. There were 42875 collision checks which took 242s, and the solution computation took 13 minutes.

Figure 11 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. There were 94249 collision checks which took 287s, and the solution computation took 11 minutes. Quite distinct routes, however, are taken by the robots in the different strategies. The collision region only comprises 9.89%; 83.0% of $\tilde{\mathcal{R}}$ corresponds to states in which there is only one minimal strategy. Also, 6.52% holds two solutions; 0.602% holds three solutions; 0.0265% holds four solutions; and 0.00212% holds five strategies, which is the maximum for this problem.

Figure 12 shows the minimal quotient strategies for a problem in which there are three robots that can translate or rotate along roadmaps. There were 327488 collision checks which took 38 minutes, and the solution computation took 8 hours (most of the computation is overhead due to naively processing the wavefront as a LISP list).

Figure 13 shows another "H"-shaped roadmap coordination problem; however, in this case there are three robots, and they rotate along the roadmaps. There were 425568 collision checks which took 17 minutes, and the solution computation took 14 hours. 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

This section briefly discusses an algorithm that determines one discrete-time minimal strategy on the unconstrained state space, $X = \mathcal{C}^1_{valid} \times \mathcal{C}^2_{valid} \times \cdots \times \mathcal{C}^N_{valid}$. A more thorough presentation appears in [14].

5.1 Concepts and Definitions

We first choose a vector β such that a linear scalarizing function, H, is defined using (9). As opposed to a point goal in X, we allow each robot goal to be a subset, $X^i_G \subset X^i$. We approximate (5) by discrete-time state transition equations, $x^i_{k+1} = f^i_k(x^i_k, u^i_k)$. For the computed examples that we will present, we model translation in $\Re^2$ in discrete time. 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. The state transition equation for robot $\mathcal{A}_i$ is

$$x^i_{k+1} = \left[\begin{array}{c} x^i_k[1] + \|v^i\|\Delta t \; cos(u^i_k) \\ x^i_k[2] + \|v^i\|\Delta t \; sin(u^i_k) \end{array} \right]. \quad (17)$$

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

$$L^{i*}_k(x_k) = \sum_{k'=k}^{K} \left\{ l^i_{k'}(x^i_{k'}, u^i_{k'}) + \sum_{j \neq i} c^{ij}_{k'}(x(\cdot)) \right\} + q^i(x^i_{K+1}). \quad (18)$$

The function $L^{*i}_k(x_k)$ is sometimes referred to as the *cost-to-go* function in dynamic optimization literature. For this context, we modify the definition of $q^i(x^i_{K+1})$ in (6), 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 $H(\gamma, \beta)$ (from (9)) to obtain H^*_k, which represents a single cost-to-go function.

The principle of optimality 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:

$$H_k^*(x_k) = \min_{u_k \in U} \left\{ \sum_{i=1}^{N} \beta_i \, l_k^i(x_k, u_k) + \sum_{i=1}^{N} \sum_{j \neq i} \beta_i \, c_k^{ij}(x^i(\cdot)) + H_{k+1}^*(x_{k+1}) \right\}. \tag{19}$$

For each choice of u_k, x_{k+1} is obtained by applying f_k^i for each $i \in \{1, \ldots N\}$. The boundary condition for this recurrence is given by

$$H_{K+1}^* = \sum_{i=1}^{N} \beta_i q^i(x_{K+1}^i). \tag{20}$$

We can begin with stage $K+1$, and repeatedly apply (19) to obtain the optimal actions. The cost-to-go, H_K^*, can be determined from H_{K+1}^* through (19). Using the $u_K \in U$ that minimizes (19) at x_K, we define $\gamma_K^*(x_K) = u_K$. We then apply (19) again, using H_K^* to obtain H_{K-1}^* and γ_{K-1}^*. These iterations continue until $k = 1$. Finally, we take $\gamma^* = \{\gamma_1^*, \ldots, \gamma_K^*\}$. The final cost-to-go function is essentially a global navigation function [18].

5.2 Computed Examples

We present a computed example that was obtained with the algorithm described in this section. The example involves motion planning for two robots, which are allowed to independently translate in $\Re^2$ (without restriction to a path or roadmap). For the problem in Figure 14, (9) was used with $\beta_1 = \beta_2 = \frac{1}{2}$. There were 160000 collision checks which took 17 minutes, and the solution computation took 7 hours. In the solution, neither robot is required 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 performance measures, and have presented motion planning algorithms, which were each derived from the principle of optimality. These algorithms pertain to 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 for multiple robots. Computed examples were presented for all three problem classes that illustrate the concepts and algorithms.

One useful benefit of the algorithms presented in this paper is that the minimal quotient strategies from all initial states are represented (for a fixed goal). This could be useful if we are repeatedly interested in returning the robots to some goal positions without colliding, if the initial locations vary. We could alternatively exchange the initial state and goal states in the algorithms. This would produce a representation of minimal quotient strategies to all possible goals, from a fixed initial state. This initial state can be interpreted as a "home" position for each of the robots. After running the algorithm, the robots can repreatedly solve different goals, and return to the home position by reversing the strategy.

Coordination on roadmaps provides enough maneuverability for most problems; however, in general, completeness with respect to the original problem is lost when restricted to roadmaps. Roadmaps have traditionally been determined for motion planning of a single robot, and some additional issues can be considered when constructing roadmaps for the purpose of coordination. For example, if each roadmap contains at least one configuration that is reachable by the robot, and avoids collisions with the other robots, regardless of their configurations, then general completeness is maintained. For example, we could give each robot an initial configuration in a home position or "garage", in which other robots are not allowed to enter.

Acknowledgements

Narendra Ahuja, M. Tamer Başar, Jean-Claude Latombe, Jean Ponce, Rajeev Sharma, Steve Sullivan, and Mark Spong made helpful comments and suggestions. This work was sponsored by NSF under grant #IRI-9110270, the Beckman Institute, and a Mavis Fellowship.

References

[1] M. D. Ardema and J. M. Skowronski. Dynamic game applied to coordination control of two arm robotic system. In R. P. Hämäläinen and H. K. Ehtamo, editors, *Differential Games - Developments in Modelling and Computation*, pages 118–130. Springer-Verlag, Berlin, 1991.

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

[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] A. G. Barto, R. S. Sutton, and C. J. C. H. Watkins. Learning and sequential decision making. In M. Gabriel and J.W. Moore, editors, *Learning and Computational Neuroscience: Foundations of Adaptive Networks*, pages 539–602. MIT Press, 1990.

[5] K. Basye, T. Dean, J. Kirman, and M. Lejter. A decision-theoretic approach to planning, perception, and control. *IEEE Expert*, 7(4):58–65, August 1992.

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

[7] S. J. Buckley. Fast motion planning for multiple moving robots. In *IEEE Int. Conf. Robot. & Autom.*, pages 322–326, 1989.

[8] J. F. Canny. *The Complexity of Robot Motion Planning.* MIT Press, Cambridge, MA, 1988.

[9] C. Chang, M. J. Chung, and B. H. Lee. Collision avoidance of two robot manipulators by minimum delay time. *IEEE Trans. Syst., Man, Cybern.*, 24(3):517–522, 1994.

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

[11] H. Hu, M. Brady, and P. Probert. Coping with uncertainty in control and planning for a mobile robot. In *IEEE/RSJ Int. Workshop on Intelligent Robots and Systems*, pages 1025–1030, Osaka, Japan, November 1991.

[12] K. Kant and S. W. Zucker. Toward efficient trajectory planning: The path-velocity decomposition. *Int. J. Robot. Res.*, 5(3):72–89, 1986.

[13] J.-C. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Boston, MA, 1991.

[14] S. M. LaValle. *A Game-Theoretic Framework for Robot Motion Planning.* PhD thesis, University of Illinois, Urbana, IL, July 1995.

[15] J. Miura and Y. Shirai. Planning of vision and motion for a mobile robot using a probabilistic model of uncertainty. In *IEEE/RSJ Int. Workshop on Intelligent Robots and Systems*, pages 403–408, Osaka, Japan, May 1991.

[16] 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.

[17] C. O'Dunlaing and C. K. Yap. A retraction method for planning the motion of a disc. *Journal of Algorithms*, 6:104–111, 1982.

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

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

[20] J. T. Schwartz and M. Sharir. On the piano movers' problem: III. Coordinating the motion of several independent bodies. *Int. J. Robot. Res.*, 2(3):97–140, 1983.

[21] 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.

[22] S.-H. Suh and K. G. Shin. A variational dynamic programming approach to robot-path planning with a distance-safety criterion. *IEEE Trans. Robot. & Autom.*, 4(3):334–349, June 1988.

[23] R. S. Sutton. Planning by incremental dynamic programming. In *Proceedings of the Eighth International Workshop on Machine Learning*, pages 353–357. Morgan Kaufmann, 1991.

[24] 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.

[25] 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.

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

Figure 1: For a time-invariant problem, this algorithm finds all of the minimal quotient strategies in $\tilde{\mathcal{S}}$.

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 2: Suppose that $l^i_k(x^i_k, u^i_k) = \Delta t$ for all $k \in \{1, \ldots, K\}$ and $i \in \{1, \ldots, N\}$. This algorithm finds all of the minimal quotient strategies in $\tilde{\mathcal{R}}^1 \times \tilde{\mathcal{R}}^2 \times \cdots \times \tilde{\mathcal{R}}^N$.

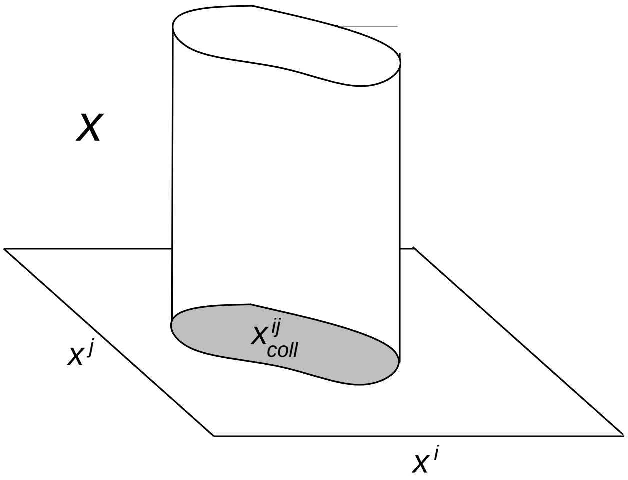
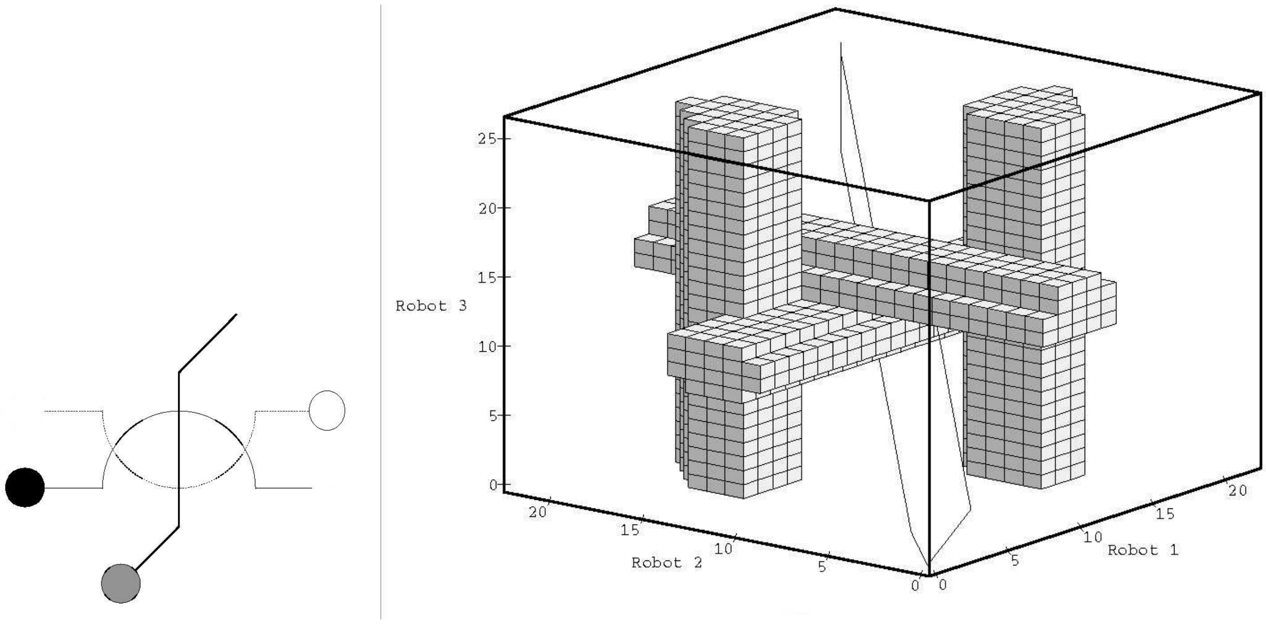


Figure 3: The set X_{coll}^{ij} and its cylindrical structure on X.



a. b.

Figure 4: A coordination space for a three-robot fixed-path problem.

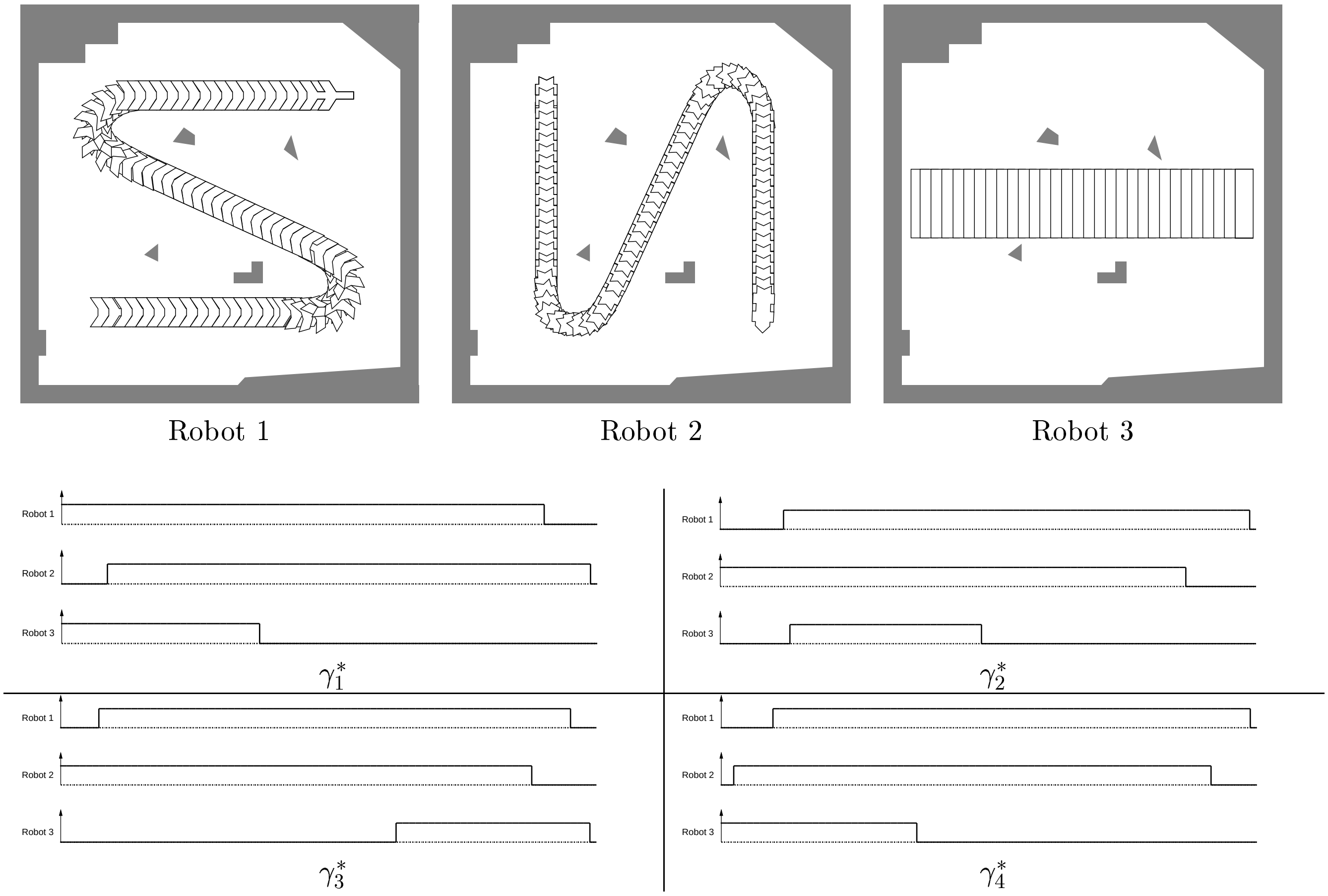


Figure 5: A problem that yields four minimal quotient strategies.

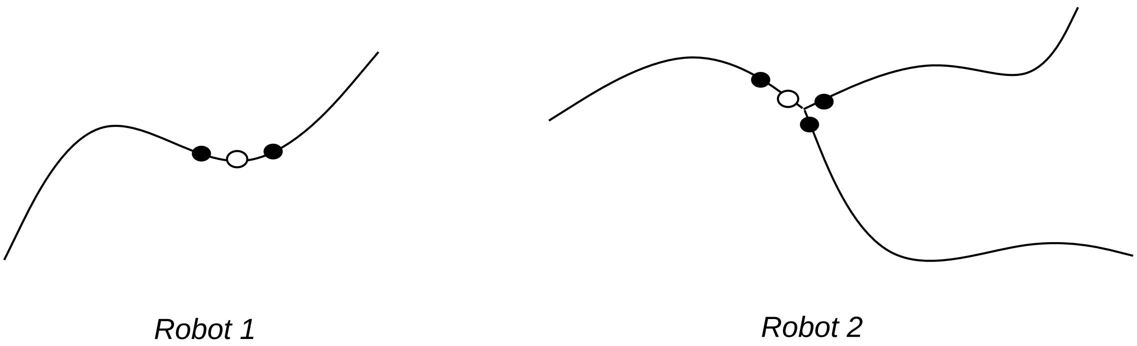


Figure 6: A two robot example in which one of the robots can make a decision about which path to continue along.

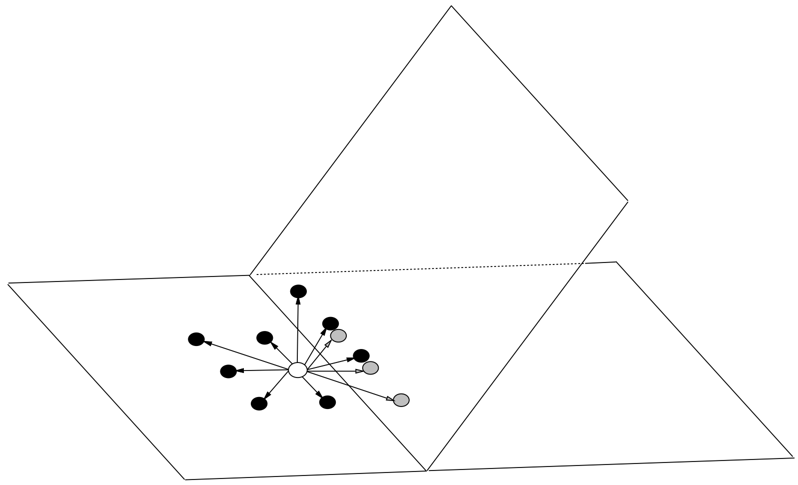


Figure 7: The corresponding path branch in the representation of $\tilde{\mathcal{R}}$.

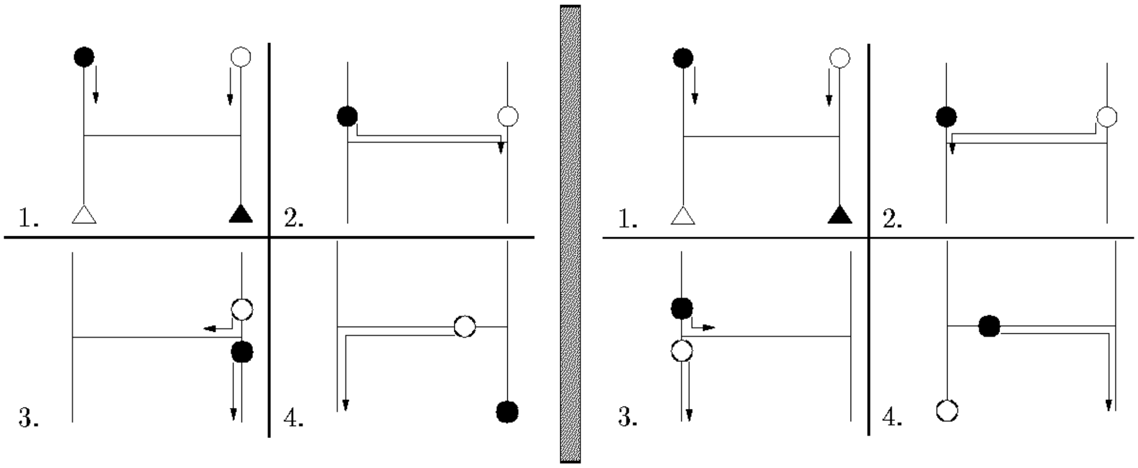


Figure 8: Two, symmetric minimal quotient strategies were computed.

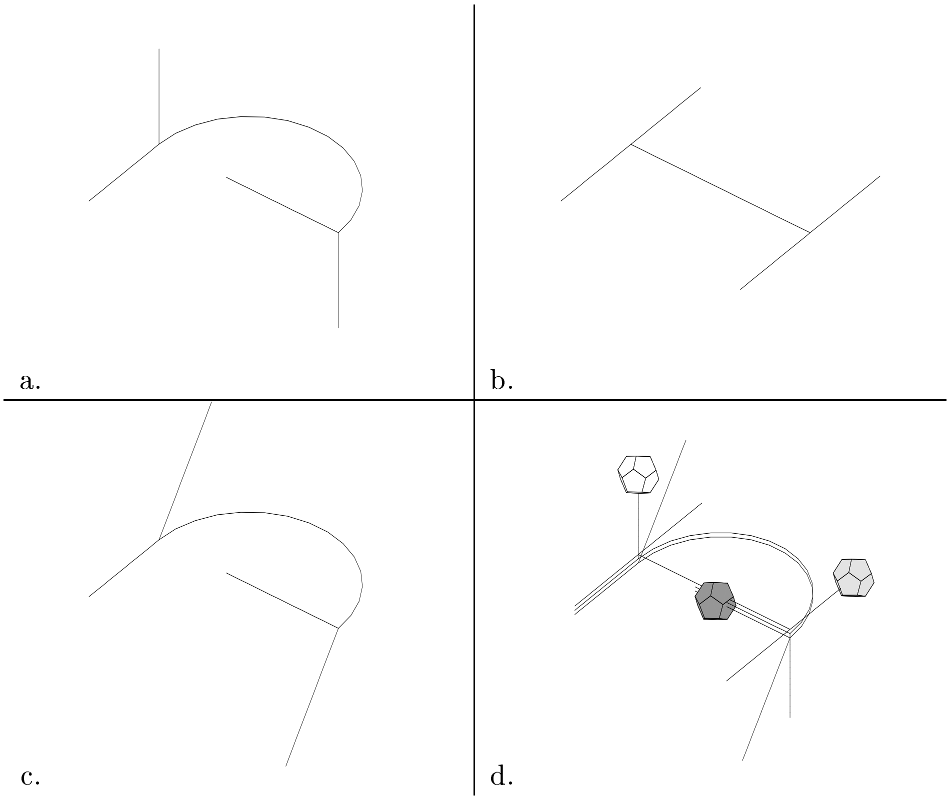


Figure 9: Parts a, b, and c show the independent roadmaps for $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{A}_3$, respectively. Part d shows the initial positions on the roadmaps.

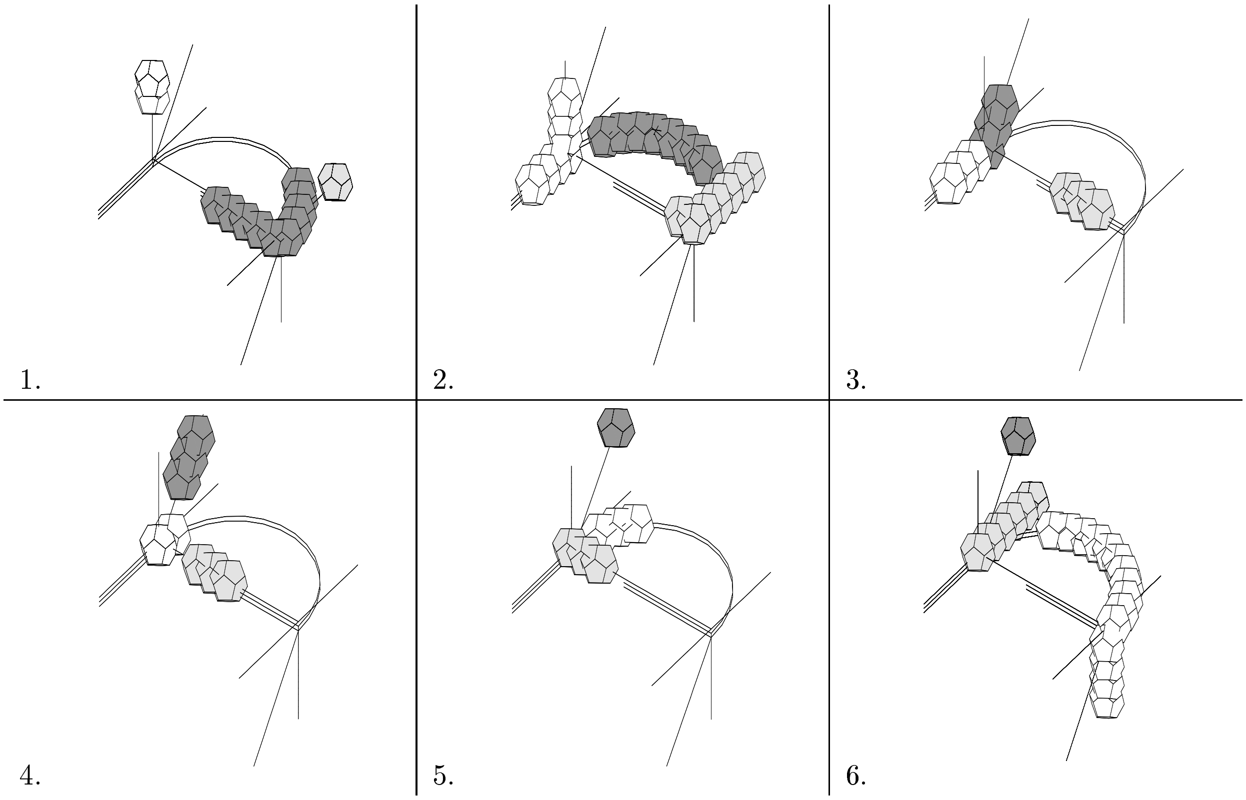


Figure 10: A representative of one of four minimal quotient strategies.

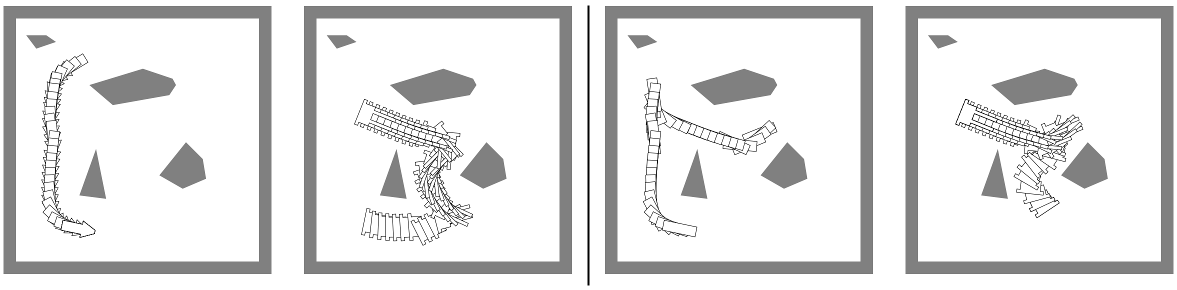


Figure 11: Two of five minimal quotient strategies for a two-robot problem with rotation.

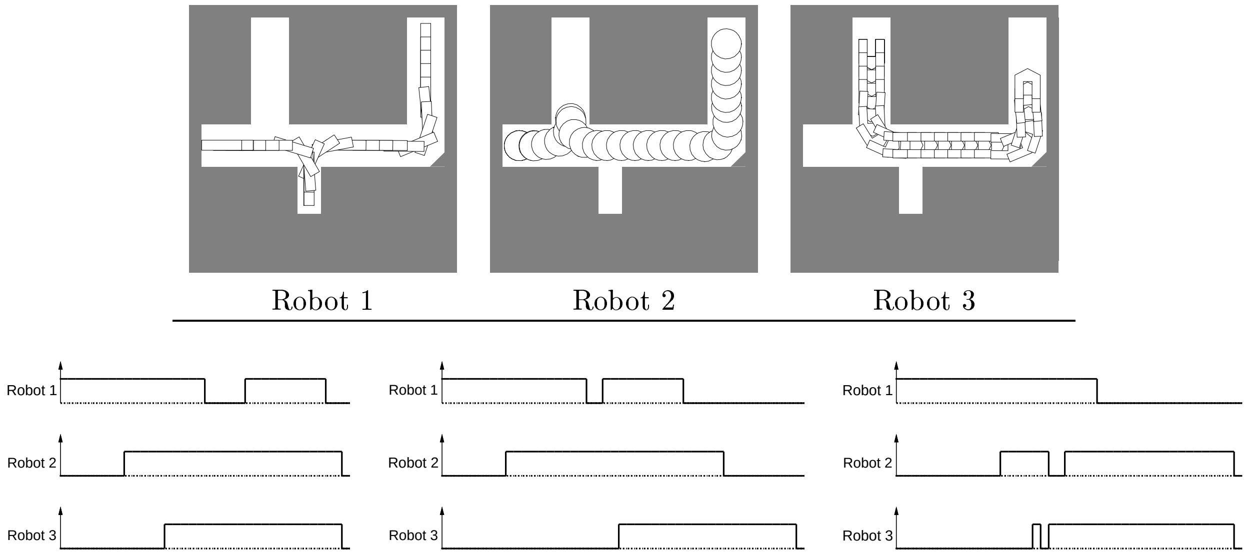


Figure 12: A problem that has three minimal quotient strategies.

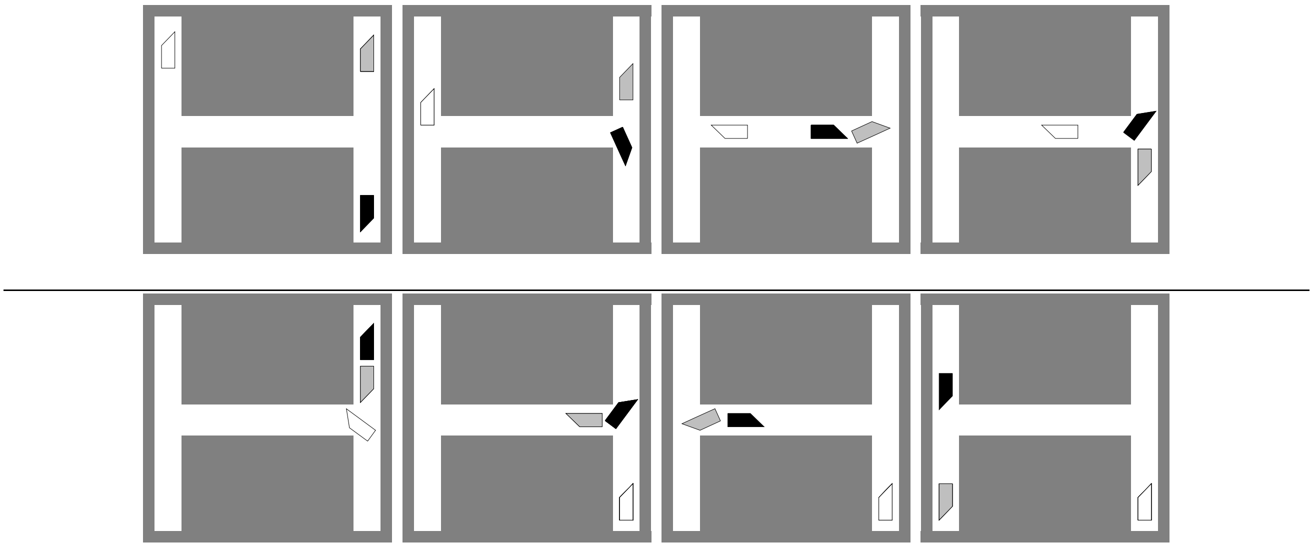


Figure 13: One solution out of sixteen is shown for three rotating robots.

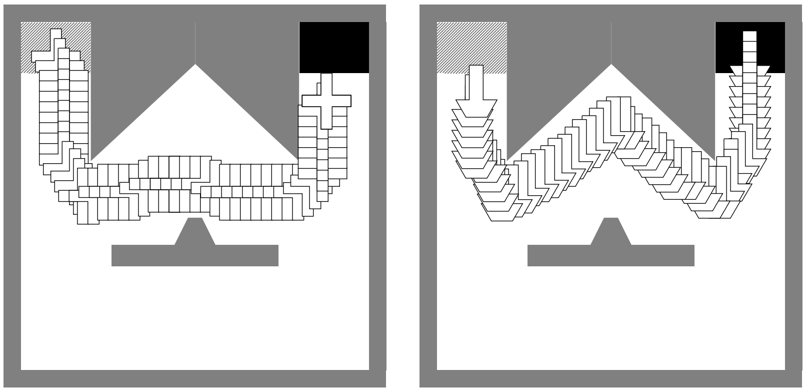


Figure 14: One representative minimal quotient strategy is given for two robots, allowed to translate in $\Re^2$.