Coordinated Path Planning for Multiple Robots

Petr Švestka, Mark H. Overmars
Department of Computer Science, Utrecht University
P.O.Box 80.089, 3508 TB Utrecht, the Netherlands
e-mail: petr@cs.ruu.nl, markov@cs.ruu.nl

Abstract

We present a new approach to the multi-robot path planning problem, where a number of robots are to change their positions through feasible motions in the same static environment. Rather than the usual decoupled planning, we use a coordinated approach. As a result we can show that the method is probabilistically complete, that is, any solvable problem will be solved within a finite amount of time. A data-structure storing multi-robot motions is built in two steps. First, a roadmap is constructed for just one robot. For this we use the *Probabilistic Path Planner*, which guarantees that the approach can be easily applied to different robot types. In the second step, a number of these simple roadmaps are combined into a roadmap for the composite robot. This data-structure can be used for retrieving multi-robot paths. We have applied the method to *car-like robots*, and simulation results are presented which show that problems involving up to 5 car-like robots in complex environments are solved successfully in computation times in the order of seconds, after a preprocessing step (the construction of the data structure) that consumes, at most, a few minutes. Such a preprocessing step however needs to be performed just once, for a given static environment.

1 Introduction

In many situations multiple mobile robots operate in the same environment. When the robots stay far enough apart they can plan their motions more or less independently, but when they get within close range of each other, their motions must be coordinated in order to avoid mutual collisions and deadlock situations. For example, when two robots want to drive though a narrow corridor in opposite directions, one of them must wait at the entrance of the corridor to let the other pass.

We present a new and powerful approach for solving multi-robot path planning problems in known static environments. In our setting, a number of robots move independently in the same workspace (containing obstacles), and the task is to compute feasible paths for the robots that bring each robot from some start configuration to some goal configuration, while avoiding (mutual) collisions. See also Figure 1.

We will refer to the separate robots $\mathcal{A}_1, \ldots, \mathcal{A}_n$ as the *simple robots.* One can also consider the simple robots together to be one robot (with many degrees of freedom), the so-called *composite robot.* A feasible path for the composite robot will be referred to as a *coordinated path.* We assume in this paper that the simple robots are identical, although, with minor adaptions, the presented concepts are applicable to problems involving non-identical robots as well.

Our approach constructs a roadmap for the composite robot, that is, we compute a network of feasible motions for the composite robot. Once this roadmap is available, path planning queries can be solved in a simple way. As a result, after the preprocessing step building the roadmap, path planning queries are solved in just a few seconds. The roadmap for the composite robot is constructed in two steps. First, a *simple roadmap* is constructed for just one robot. Then, n of such roadmaps are combined into a roadmap for the composite robot consisting of n simple robots. We will refer to such a composite roadmap as a *super-graph.* After such a super-graph has been constructed, which needs to be done just once for a given static environment, it can be used

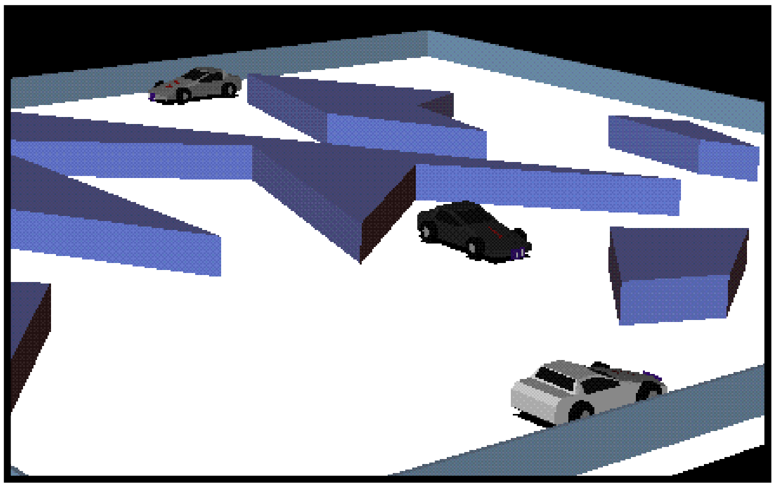


Figure 1: Simulation of 3 car-like robots moving in a cluttered environment.

for retrieval of coordinated paths. We will present two super-graph structures: *flat super-graphs* and *multi-level super-graphs*. The latter are a generalisation of flat super-graphs, and consume much less memory for problems involving more than 3 robots. The approach is a flexible one, in the sense that it is easily applicable to various robot types, provided that one is able to construct simple roadmaps for one such robot. Furthermore, proper construction of the simple roadmaps guarantees *probabilistic completeness* of the resulting multi-robot planners. In this paper we apply the super-graph approach to *car-like robots*.

The paper is organised as follows: We will first, in Section 2, give an overview on related previous work. In Section 3 we give some general definitions and notations that will be used throughout this paper. Then, in Section 4, we define formally the multi-robot path planning problem, and also we describe the discretisation that we use. The Sections 5 and 6 describe the two super-graph methods for solving multi-robot path planning problems. These methods are applied to problems involving up to 5 car-like robots in Section 8, and simulation results are presented. These demonstrate the methods power and its efficiency regarding computation costs and memory consumption. In Section 9 we discuss how a form of probabilistic completeness of the multi-robot planners can be obtained, in the case where the simple roadmaps are constructed by the Probabilistic Path Planner *PPP* [OŠ94, ŠO95b, KŠLO96, Kav95]. Finally, some conclusions are drawn and some future work is indicated in Section 10.

2 Related previous work

Motion planning for multiple robots has been studied extensively over the past ten years. We review some major results. (For more detailed overviews see also [Lat91] and [HA92].) Approaches to the multi-robot motion planning problem are often categorised as *centralised* and *decoupled*. Centralised approaches treat the separate robots as one composite system, and typically perform the planning in a composite configuration space, formed by combining the configuration spaces of the individual robots. Decoupled approaches first generate paths for the separate robots more or less independently, and then consider the interactions between the robots (with respect to the generated paths).

Centralised approaches have the advantage that, at least in theory, they allow for *complete* planners (that always find a solution when one exists), whereas decoupled planners inherently are not complete and can lead to deadlock situations. However, the practical difficulty with centralised planning is that, if completeness is to be obtained, it yields methods whose time complexity is exponential in the dimension of the composite configuration space. For example, if we have m robots with k degrees of freedom each, a complete planner will have a time complexity exponential in $m*k$. This means that, assuming we do not consider the dimension of the composite configuration space as a constant, the multi-robot path planning problem is *PSPACE-complete* (See, e.g., [Lat91]). Even the apparently simple problem of motion planning for arbitrarily many rectangular robots in an empty rectangular workspace is still PSPACE-complete [HSS84].

Centralised planning Schwartz and Sharir [SS83] have described an exact cell decomposition algorithm for multiple discs in the plane moving among polygonal obstacles. Their approach is based on *critical curves*. The complexity of the algorithm is $O(n^3)$ for 2 discs, and $O(n^{13})$ for 3 discs. In [SS88], Sharir and Sifrony present a general $O(n^2)$ algorithm for two robots each with 2 degrees of freedom. The robots can be discs or manipulators. Ardema and Skowronski [AS91] describe a method for generating collision-free motion control for two specific, constrained manipulators, by modelling the problem as a non-cooperative game. Other centralised approaches for tackling multi-robot planning problems include various *potential field* techniques. Tournassoud [Tou86] proposes a potential field approach where the motion coordination is expressed as a local optimisation problem. Barraquand and Latombe [BL90, BL91] have applied the *randomised path planner RPP*, a general potential field method that uses random motions for escaping local minima, to multi-robot planning problems. In [BLL92] Barraquand, Langlois, and Latombe present a potential field technique for many discs in environments with narrow corridors. To escape local minima, so-called *constrained motions* are executed which force one configuration coordinate to increase or decrease until a saddle point of the potential function is attained. This potential field planner has been successfully experiment ed with for up to 10 robots.

Decoupled planning Erdmann and Lozano-Pérez [ELP86] proposed the scheme of *prioritised planning*. First, priorities are assigned to the robots. Then, in order of decreasing priority, the robots are picked. For each picked robot a path is planned, avoiding collisions with the static obstacles as well as the previously picked robots, which are considered as moving obstacles. An important question in this scheme is how to assign the different priorities to the robots. Buckley [Buc89] proposes to assign the priorities such that the number of robots that can move in a straight line from their start configuration to their goal configuration (so-called *linear robots*) is maximised. In [War90], prioritisation is combined with a potential field that is defined on a time-varying configuration space.

O'Donnell and Lozano-Pérez [OLP89] introduced a decoupled planning approach called *path coordination*, which is based on a scheduling technique originally developed by Yannakakis et al. [YPK79] for concurrent database access by several users. It is assumed that the planning problem involves just two robots. First, paths P_1 and P_2 for the two robots are planned independently. Then, a two dimensional coordination diagram is constructed, where the x-axis corresponds to path P_1 and the y-axis to path P_2. A point (x, y) in the coordination diagram corresponds to the placement of the first robot at the configuration $P_1(x)$ and the second robot at the configuration $P_2(y)$[1]. If the two robots intersect at the configurations $P_1(x)$ and $P_2(y)$, then (x, y) is a forbidden point in the coordination diagram. The path coordination now is performed by searching for a path connecting the bottom left corner and the top right corner in the coordination diagram, not going through any forbidden points. If it is found, such a path describes a velocity coordination of the two robots along the two paths that brings both robots to their goal configurations without mutual collisions. Whether such a coordination exists of course depends on the initially chosen paths P_1 and P_2. Coordination diagrams for two manipulators are analysed in [BL92, CCL94]. Furthermore, scheduling issues are studied through use of coordination diagrams in [LT90].

Another decoupled approach has been developed by Liu et al. [LKN+89]. Here, as in the path coordination approach, the initial path of each robot is planned separately. Then intersections among the paths are tested for. If these exist, it is attempted to resolve the conflicts by globally

[1]Given a path P for some robot, we denote the configuration that the robot attains at time t when moving along the path by $P(t)$. This is formalised in Section 3.

modifying the initial paths using a Petri net formulation. In concept the planner is applicable to systems with many robots. The presented implementation however only works for a two robot system.

Alami et al. propose a *Plan Merging Paradigm* [ARIS95] for execution and control of a large fleet of autonomous mobile robots. The robots incrementally merge their plans into a set of already coordinated plans, through exchange of information about their current state and their future actions. Planners based on traffic rules are described in [Gro88, KNT92, Wan95, WP95, LLC$^+$95]. Some recent work on decoupled planners experimented with on real robots includes [CE92, CFA$^+$95, SG96]. As is the case in general for decoupled methods, these planners do not work well in very constrained environments.

Weaker centralisation: Roadmap coordination As mentioned above, in order to obtain complete planners, centralised planning approaches are required. For relatively simple problem settings, some practical complete planners have been developed/proposed (see above). However, it seems that for more challenging problems the centralised approach leads to prohibitive time complexity due to the PSPACE-completeness of the multi-robot planning problem. For this reason many researchers have dropped the requirement of completeness and focused on decoupled approaches. This has resulted in many planners that solve interesting problems, but satisfy no form of completeness. As also pointed out by La Valle and Hutchinson in [VH95], it seems that most research has focused on two opposite ends of a spectrum, with centralised planning at one end and decoupled planning at the other. With centralised planning, no constraints are imposed on the robot motions before considering the interactions between the robots, while with decoupled planning very strong constraints are imposed on the robot motions prior to considering the robot interactions. An approach lying somewhere in the middle of the spectrum will be one that only weakly constrains motions of the separate robots. This can be achieved by constraining the robots to lie on independent roadmaps (instead of paths). This *roadmap coordination* idea has, in different forms, previously been used by Shibita and Fukuda in [SF93] and by La Valle and Hutchinson in [VH95].

The super-graph scheme that we propose in this paper[2] falls in the class of roadmap coordination methods. Roadmaps are constructed for the separate robots, and the robots are constrained to lie on these. The coordination of the robots along these roadmaps is achieved by building and searching a data-structure that, in concept, represents a Cartesian product of the separate roadmaps. A strong point of our approach is that it leads to *probabilistically complete* planners that are capable of solving complicated problems that cannot be solved by decoupled planners and would consume far too much time and memory for existing complete planners. This shows that indeed by moving away from the extremes on the "centralisation spectrum" one can obtain forms of completeness while yielding planners of acceptable time-complexity.

3 Preliminary definitions and notations

We start with giving some definitions and notations that will be used throughout this paper.

The robots and their C-space Let $\mathcal{A}_1, \ldots, \mathcal{A}_n$ be n instances of some (simple) robot $\mathcal{A}$, present in a workspace $\mathcal{W}$, together with a set of obstacles whose union we denote by $\mathcal{B}$. We refer to $\mathcal{A}$ as the *simple robot*, and to $(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ as the *composite robot*. Furthermore, let $\mathcal{C}$ be the space of all possible configurations of $\mathcal{A}$, and let $\mathcal{C}_f$ be a the subset of $\mathcal{C}$ consisting of all configurations c such that $\mathcal{A}$ placed at c intersects no obstacles. That is, $\mathcal{C}_f$ is $\mathcal{A}$'s free configuration space, or free *C-space*. Given a configuration c, we denote the workspace-area occupied by $\mathcal{A}$, when placed at c, by $\mathcal{A}(c)$.

[2]Preliminary extended abstracts of this paper have been given in [ŠO95a, ŠO96]

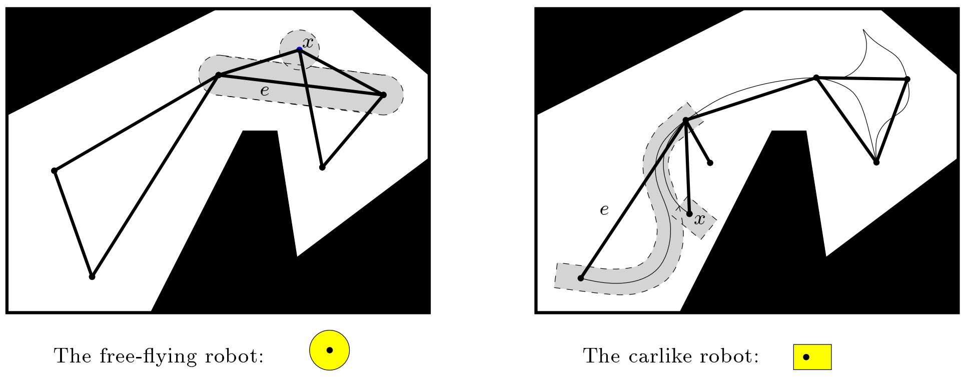


Figure 2: Two simple roadmaps. At the left the robot is a translating disc, and at the right a car-like rectangle. The edges are shown in thick lines. In the left roadmap the actual local paths coincide with the edges. In the right roadmap this is not the case, and the corresponding local paths are indicated by thin curves. In both examples we see that an edge e and a node x with $\mathcal{A}(e) \otimes \mathcal{A}(x)$ (because e's sweep-volume intersects $\mathcal{A}$ placed at x). This means that x blocks e.

Paths and roadmaps for the simple robots Now let us look at *paths*. A *path* P for $\mathcal{A}$ is a continuous map $P \in [0,1] \to \mathcal{C}$. For $[0,1] \to \mathcal{C}$, we also use the shorthand notation $\mathcal{C}^{[0,1]}$. It maps time t to configurations, hence describing a *motion* of the robot $\mathcal{A}$. If P lies in $\mathcal{C}_f$ (that is, $\forall t \in [0,1] : P(t) \in \mathcal{C}_f$), then we refer to P as a *free path*. If, in addition, P describes motions that are performable by the robot $\mathcal{A}$, we say that P is a *feasible path* for $\mathcal{A}$. A path that describes no motion of $\mathcal{A}$ (that is, $\mathcal{A}$ is stationary), we refer to as an *identity path*. For this we define $I \in \mathcal{C} \to \mathcal{C}^{[0,1]}$ as the function mapping c to $t \mapsto c$. We denote the workspace-area swept by $\mathcal{A}$, when moving along a path P, by $\mathcal{A}(P)$. Given two paths P and Q with $P(1) = Q(0)$, we denote their concatenation by $P \oplus Q$. Formally:

$$(P \oplus Q)(t) = P(2t) \text{ for } t \leq \frac{1}{2}, \text{ and } Q(2t-1) \text{ for } t > \frac{1}{2}$$

We will use roadmaps for the simple robots (*simple roadmaps*) as basis for building the super-graphs. For this, we assume the simple roadmap of $\mathcal{A}$ to be of a certain form. Namely, we assume it to be stored as a graph $G = (V, E)$, with the nodes V corresponding to collision-free configurations. The edges are pairs of collision-free configurations and correspond to feasible paths. We assume namely that we have a *local planner* $L \in \mathcal{C} \times \mathcal{C} \to \mathcal{C}^{[0,1]}$ that constructs paths that are feasible for $\mathcal{A}$ in absence of obstacles (that is, paths describing motions that are performable by $\mathcal{A}$), and an edge (a, b) is allowed to be present in E only if $L(a, b)$ is collision-free (and, hence, feasible). Given an edge $(a, b) \in E$, we refer to $L(a, b)$ as the *local path* connecting a and b. So, given a path $[c_1, c_2, \ldots, c_k]$ in the graph G, $L(c_1, c_2) \oplus L(c_2, c_3) \oplus \ldots \oplus L(c_{k-1}, c_k)$ is a feasible path for $\mathcal{A}$, from c_1 to c_k. The workspace area swept by $\mathcal{A}$ when moving along a local path $L(e)$ is denoted by $\mathcal{A}(e)$, e's *sweep volume*. For convenience, for arbitrary objects A and B, we will refer to the statement $A \cap B \neq \emptyset$ (A and B intersect) by $A \otimes B$. We say that an edge $e \in E$ is *blocked* by a node $x \in V$ if $\mathcal{A}(e) \otimes \mathcal{A}(x)$ (see also Figure 2). An edge e being blocked by a node x means that a simple robot cannot move along the local path corresponding to e if another simple robot is stationary at the configuration x.

Paths for the composite robot A *coordinated path* for the composite robot $(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ is a n-tuple of paths feasible for $\mathcal{A}$ that, when executed simultaneously, introduce no mutual collisions between the simple robots. This is formalised by Definition 1.

Definition 1 *Let $\mathcal{A}_1, \ldots, \mathcal{A}_n$ be n simple robots, and let $s_1, \ldots, s_n$ and $g_1, \ldots, g_n$ be given free configurations (that is, $\forall i \in \{1, \ldots n\} : s_i \in \mathcal{C}_f \wedge g_i \in \mathcal{C}_f$). If $P_1, \ldots, P_n \in \mathcal{C}^{[0,1]}$ are feasible paths for $\mathcal{A}$, such that for all $i, j \in \{1, \ldots, n\}$*

- $P_i(0) = s_i \wedge P_i(1) = g_i$
- $i \neq j \Rightarrow \forall t \in [0, 1] : \neg \mathcal{A}(P_i(t)) \otimes \mathcal{A}(P_j(t))$

then $(P_1, \ldots, P_n)$ is a coordinated path *for $(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ solving the problem $((s_1, \ldots, s_n), (g_1, \ldots, g_n))$. We define the* length *of $(P_1, \ldots, P_n)$ as $\sum_{i=0}^{n}$ length(P_i).*

Given two coordinated paths $\mathcal{P} = (P_1, \ldots, P_n)$ and $\mathcal{Q} = (Q_1, \ldots, Q_n)$ with $\mathcal{P}(1) = \mathcal{Q}(0)$, we denote their concatenation $(P_1 \oplus Q_1, \ldots, P_n \oplus Q_n)$ by $\mathcal{P} \oplus \mathcal{Q}$. Clearly, $\mathcal{P} \oplus \mathcal{Q}$ will be a coordinated path as well.

A special type of coordinated paths that we will use as basic building blocks is that of *trivial coordinated paths.* A *trivial coordinated path* is defined as a coordinated path $(P_1, \ldots, P_n)$ where at most one P_i is not an identity path. That is, such a path describes a multi-robot motion where at most one of the simple robots moves, while all the others are stationary. A *discretised coordinated path* is now defined as being the concatenation of a finite number of trivial coordinated paths. So a discretised coordinated path describes a multi-robot motion where at any time instant at most one robot moves.

Definition 2 *A coordinated path for $(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ is a* discretised coordinated path *if it is the concatenation of a finite number of trivial coordinated paths.*

4 Discretising the multi-robot planning problem

Definition 3 *The multi-robot path planning problem for $\mathcal{A}_1, \ldots, \mathcal{A}_n$ is defined as follows: Given start configurations $s_1, \ldots, s_n$ and goal configurations $g_1, \ldots, g_n$ (with $s_i, g_i \in \mathcal{C}_f$), find a* coordinated path *for $(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ solving the problem $((s_1, \ldots, s_n), (g_1, \ldots, g_n))$*

The first step of our multi-robot planning scheme consists of computing a simple roadmap, that is, a roadmap for the simple robot $\mathcal{A}$. This roadmap will subsequently be used for the construction of a roadmap for the composite robot. Basically, any algorithm that constructs roadmaps can be used in this phase. However, in order to obtain some form of completeness, there are certain requirements. In Section 9 we give a condition that guarantees a form of probabilistic completeness of the multi-robot scheme when the Probabilistic Path Planner (Section 8.2) is used for construction of the simple roadmaps.

Given a graph $G = (V, E)$ storing a simple roadmap for robot $\mathcal{A}$, we are interested in solving multi-robot problems using G. We assume here, for the moment, that all start configurations s_i and goal configurations g_i are nodes of G (We drop this assumption in the next section.). The idea is that we seek paths in G along which the robots can go from their start configurations to their goal configurations, such that at any time instant:

- at most 1 robot moves along a local path $L(e)$,
- the other robots are stationary at nodes $x_1, \ldots, x_{n-1}$ of G,
- and none of the nodes $x_1, \ldots, x_{n-1}$ block the edge e.

In this way, clearly, mutual robot collisions are avoided, while robot-obstacle collisions are ruled out by the fact that we move along the simple roadmap. We say that we *discretise* the multi-robot planning problem to G.

Definition 4 *Let $G = (V, E)$ be a simple roadmap. A coordinated path for $(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ is a* G-discretised coordinated path *if it is the concatenation of a finite number of trivial coordinated paths $T_1, \ldots, T_k$, where* $(\forall i \in \{1, \ldots, k\})$

$$T_i = (I(c_1), \ldots, I(c_m), L(e), I(c_{m+1}), \ldots, I(c_{n-1})) \text{ with the } c_j\text{'s in } V \text{ and } e \in E$$

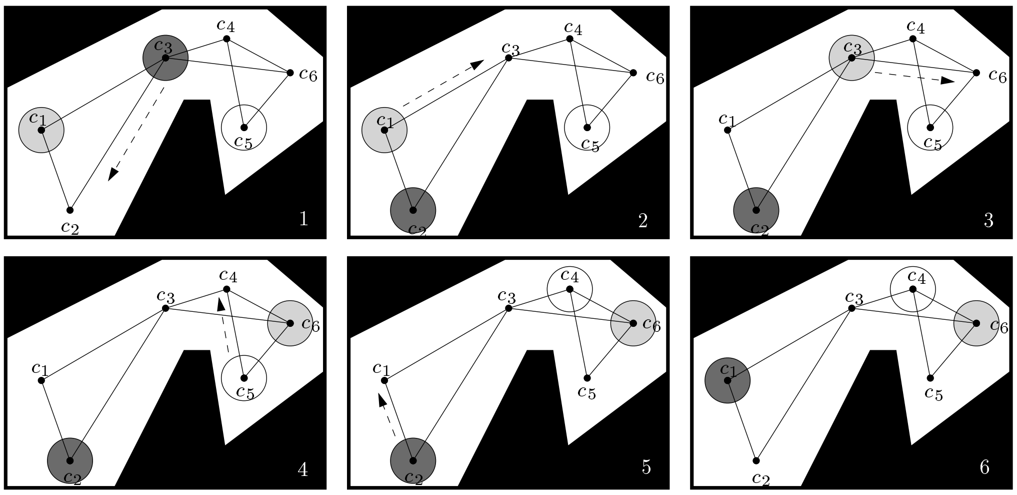


Figure 3: A G-discretised coordinated path $\mathcal{P}$ for 3 translating disc-robots, solving the problem $((c_5, c_1, c_3), (c_4, c_6, c_1))$. If we denote the edge connecting nodes c_i and c_j by e_{ij}, then $\mathcal{P} = (I(c_5), I(c_1), L(e_{32})) \oplus (I(c_5), L(e_{13}), I(c_2)) \oplus (I(c_5), L(e_{36}), I(c_2)) \oplus (L(e_{54}), I(c_6), I(c_2)) \oplus (I(c_4), I(c_6), L(e_{21}))$.

See Figure 3 for a simple example of a G-discretised coordinated path. Of course, moving only one simple robot at a time will result in unnecessarily long paths for the composite robot, but, as we shall see later, this can be remedied with some simple smoothing techniques. In Section 9 we show that, provided a suitable roadmap construction method is used, solving G-discretised path planning problems (instead of continuous ones) is sufficient to guarantee *probabilistic completeness* of the multi-robot planning scheme.

5 The flat super-graph method

The question now is, given a simple roadmap $G = (V, E)$ for a robot $\mathcal{A}$, how to compute G-discretised coordinated paths for the composite robot $(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ (with all $\mathcal{A}_i$ identical to $\mathcal{A}$).

For finding G-discretised coordinated paths, we introduce the notion of *flat super-graphs*:

Definition 5 *Let $G = (V, E)$ be a simple roadmap and let n be the number of simple robots. The induced* flat super-graph $\mathcal{F}^n_G = (V_\mathcal{F}, E_\mathcal{F})$ *is defined as follows:*

- *$(x_1, \ldots, x_n) \in V^n$ is a node of $\mathcal{F}^n_G$ if and only if $i \neq j \Rightarrow \neg\mathcal{A}(x_i) \otimes \mathcal{A}(x_j)$.*
  We refer to the nodes of $\mathcal{F}^n_G$ as (flat) super-nodes. *Given a super-node $X = (x_1, \ldots, x_n)$, we refer to the x_i's as X's* underlying G-nodes.
- *Two super-nodes $X = (x_1, \ldots, x_n)$ and $Y = (y_1, \ldots, y_n)$ are connected by an edge of $\mathcal{F}^n_G$ if and only if for exactly one $i \in \{1, \ldots, n\}$ $x_i \neq y_i$, and x_i is connected to y_i by an edge e in G that is not blocked by any x_j, with $j \neq i$. The edge e is referred to as (X, Y)'s* underlying G-edge. *We refer to the edges of $\mathcal{F}^n_G$ as* (flat) super-edges.

We refer to G as the underlying simple roadmap *of $\mathcal{F}^n_G$.*

See Figure 4 for an example of a (simple) flat super-graph. So each node of $\mathcal{F}^n_G$ corresponds to a feasible placement of the n simple robots at nodes of G, and each edge of $\mathcal{F}^n_G$ corresponds to a feasible motion of one simple robot along an edge of G. That is, each edge in $\mathcal{F}^n_G$ corresponds to a trivial coordinated path. Hence, any path in the flat super-graph describes a G-discretised

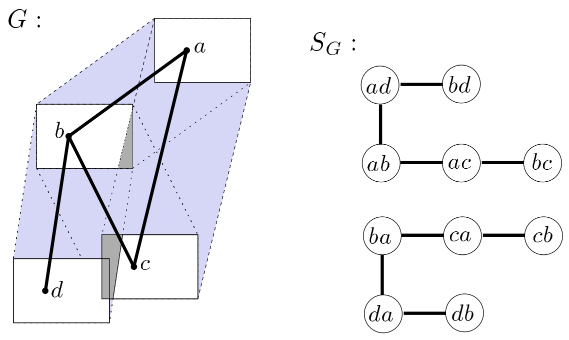


Figure 4: At the left we see a simple roadmap G for the shown rectangular robot $\mathcal{A}$ (shown in white, placed at the graph nodes). We assume here that $\mathcal{A}$ is a translational robot, and the areas swept by the local paths corresponding to the edges of G are indicated in light grey. At the right, we see the flat super-graph $\mathcal{F}_G^2$, induced by G. It consists of two separate connected components.

coordinated path, and vice-versa. So we see that the problem of finding G-discretised coordinated paths for our composite robot reduces to graph searches in $\mathcal{F}_G^n$.

We however want to use the flat super-graph for solving arbitrary multi-robot problems, not only such where the start and goal configurations are nodes of G. For this, we define what we refer to as *coordinated retractions*.

Definition 6 *Let* $G = (V, E)$ *be a simple roadmap. A coordinated retraction of a tuple* $(c_1, \ldots, c_n) \in \mathcal{C}_f^n$ *is a coordinated path* $(P_{c_1}, \ldots, P_{c_n})$ *with:* $\forall i \in \{1, \ldots, n\} : P_{c_i}(0) = c_i \wedge P_{c_i}(1) \in V$.

In words, a coordinated retraction of $(c_1, \ldots, c_n)$ is a coordinated path, moving each simple robot $\mathcal{A}_i$ from c_i to a node of G. Of course, coordinated retractions are easy to find only if G is of sufficient density. The algorithm for the computation of the retractions depends on the type of roadmap used.

Algorithm 1 now summarises how a multi-robot problem can be solved with a flat super-graph.

Algorithm 1 – Flat super-graph method

Let G *be a simple roadmap, and* $\mathcal{F}_G^n$ *the induced flat super-graph. Let* $((s_1, \ldots, s_n), (g_1, \ldots, g_n))$ *be a problem to be solved.*

1. *Compute a coordinated retraction* $(P_{s_1}, \ldots, P_{s_n})$ *for* $(s_1, \ldots, s_n)$. *(If this fails, then* G *is of insufficient density.)*

2. *Compute a coordinated retraction* $(P_{g_1}, \ldots, P_{g_n})$ *for* $(g_1, \ldots, g_n)$. *(If this fails, then* G *is of insufficient density.)*

3. *Find the shortest path* $P_{\mathcal{F}}$ *in* $\mathcal{F}_G^n$ *between node* $(P_{s_1}(1), \ldots, P_{s_n}(1))$ *and node* $(P_{g_1}(1), \ldots, P_{g_n}(1))$. *If no such path exists, then there exists no* G*-discretised solution to the retracted problem.*

4. *Transform the graph-path* $P_{\mathcal{F}}$ *to a* G*-discretised coordinated path* $\mathcal{P}$. *This can be done by transforming each pair of consecutive super-nodes in* $P_{\mathcal{F}}$ *to a trivial coordinated path (using the functions* I *and* L*), and concatenating the resulting coordinated paths.*

5. *Let* P *be the concatenation of* $(P_{s_1}, \ldots, P_{s_n})$, $\mathcal{P}$, *and* $(P_{g_1}, \ldots, P_{g_n})$ *reversed.*

6. *Smooth* P.

Typically, the paths directly retrieved from the super-graph are unnecessarily long. Partially this can be caused by the underlying simple roadmap. For example, in our implementation (as described in Section 8) we will use *probabilistic* graphs as underlying simple roadmaps, forcing the simple robots to follow routes between randomly generated configurations. Furthermore, we obtain G-discretised coordinated paths from the super-graph. As explained earlier, these describe multi-robot motions where only one simple robot moves at a time. Clearly, to reduce the total path length, it is often beneficial to allow for simultaneous robot motions. This is why, in the last step of Algorithm 1, we *smooth* the coordinated path. The aim of this post-processing step is to reduce the coordinated path length, by short-cutting redundant path segments and combining alternating simple robot motions into simultaneous ones. In Section 7 we will present a general heuristic technique for this smoothing.

The size of a flat super-graph, as defined above, is exponential in n (the number of robots). However, the entire data-structure does not have to be stored explicitly. Given a particular super-node X, its neighbours in $\mathcal{F}_G^n$ can each be retrieved in constant time provided that, for each $(x, e) \in V \times E$, we know whether $\mathcal{A}(x)$ intersects $\mathcal{A}(e)$. This asks for a data-structure of quadratic size (in the size of G) that for each node-edge pair (x, e) stores whether $\mathcal{A}(x) \otimes \mathcal{A}(e)$. Using optimised intersection routines, such a data-structure, which we refer to as *the G-intersection map*, can be computed and updated quite efficiently. Hence, for performing graph searches in $\mathcal{F}_G^n$, we need only to compute and store the set $V_{\mathcal{F}}$ of super-nodes. If however n is large, then the required amount of memory can still be very (too) large. For example, if the underlying simple roadmap has 100 nodes and there are 4 robots, then the induced flat super-graph might contain up to $100^4 = 100.000.000$ nodes. Our experiments indeed show that for up to 3 robots the flat super-graph method works fine, but for more robots the memory consumption becomes a bottleneck. So we should try to reduce the number of super-nodes. The *multi-level super-graph method*, described in the next section, aims at this.

6 The multi-level super-graph method

A way to reduce the size of the super-graph is to combine multiple node-tuples into single super-nodes. We will first describe and motivate the idea informally in Section 6.1. In Section 6.2 we define the algorithm and the required data-structures. Section 6.3 deals with a technique for further reducing the size of the data-structure.

6.1 The idea

When humans are driving, and they know that no other vehicles are near them, they do not take into account the exact positions of those other vehicles for every maneuver they perform. Only as other vehicles come within closer range, more care is taken to avoid collisions. And only in really pathetic situations do drivers actually have to get out of their cars, and start talking to each other in order to make a “coordinated plan”.

In other words, as long as the cars are at safe distances from each other, the maneuvers are performed more or less locally. Now because in the super-graph data-structure no distinction is made between situations where the robots are located near to each other, and situations where this is not the case, it intuitively seems that a flat super-graph stores too much information. One would expect that size reduction of our data-structure should be possible by avoiding the storage of super-nodes corresponding to robot placements where the robots lie far apart from each other.

As explained, we plan the robot motions along local paths stored in a simple roadmap, and no direct information about the workspace and C-space is used. Hence, we must define the notion of two robots being “near” to each other in terms of the simple roadmap structure. The idea is to consider, for each simple robot, certain neighbourhoods (subgraphs) within G in which the robot is present, and to say that two robots are “near” to each other if, while staying within their neighbourhoods (subgraphs), they could block each others motions. During the planning, we will, at each planning step, identify maximal robot neighbourhoods (subgraphs) such that the

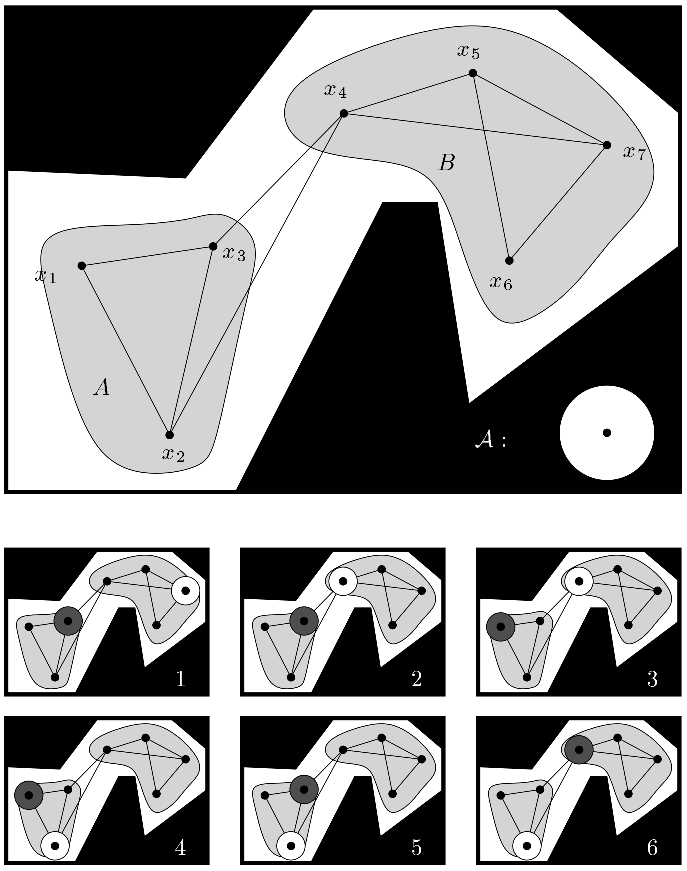


Figure 5: A subdivision of the simple roadmap into two subgraphs A and B. Combining the subgraphs into super-nodes reduces the size of the super-graph.

robots are not "near" to each other. In this way it will suffice to keep track of the occupied neighbourhoods, and it will allow us to ignore the exact robot positions in the simple roadmap.

Let us clarify the idea with an example. See Figure 5. We see a simple roadmap $G = (V, E)$ for the shown free-flying disc-like robot. Let us assume that we want to use it for finding G-discretised coordinated paths for two such robots $\mathcal{A}_1$ and $\mathcal{A}_2$. For this we can build the induced flat super-graph $\mathcal{F}_G^2 = (V_\mathcal{F}, E_\mathcal{F})$, where $V_\mathcal{F}$ will be $V \times V$. The problem $((x_7, x_3), (x_2, x_4))$ is solved by the G-discretised coordinated path corresponding to the path $[(x_7, x_3), (x_4, x_3), (x_4, x_1), (x_2, x_1), (x_2, x_3), (x_2, x_4)]$ in $\mathcal{F}_G^2$. As explained in the previous section, this path is present in the flat super-graph $\mathcal{F}_G^2$.

Now let us consider a subdivision of G into two connected subgraphs A and B (induced by, respectively, $\{x_1, x_2, x_3\}$ and $\{x_4, x_5, x_6, x_7\}$) as shown in the figure. No node in either of the two subgraphs blocks an edge in the other subgraph. Hence if $\mathcal{A}_1$ and $\mathcal{A}_2$ are positioned at nodes in, respectively, A and B, either of the robots can move along any local path within its subgraph, without possibly colliding with the other (stationary) robot. We say that A and B are *independent* (This notion is formalised in Section 6.2.). The idea now is that we replace the set of flat super-nodes $\{x_1, \ldots, x_3\} \times \{x_4, \ldots, x_7\}$ by just one super-node (A, B), and the set of flat super-nodes $\{x_4, \ldots, x_7\} \times \{x_1, \ldots, x_3\}$ by just one super-node (B, A). Through this we achieve a significant size reduction of our super-graph.

Let us again consider the problem $((x_7, x_3), (x_2, x_4))$. The tuples (x_7, x_3) and (x_2, x_4) are now not longer present as nodes in the super-graph, but instead we have the super-nodes (A, B) and (B, A) that "contain" (x_7, x_3) and (x_2, x_4). Hence, we seek a path in our "reduced" graph solving this problem. For example, (A, B) and (B, A) will be connected by the path $[(A, B), (x_2, x_1), (x_2, x_3), (B, A)]$. We will give the exact definition of the data-structure in the next section. Also we will show that any G-discretised coordinated path will correspond to a path in the "reduced" super-graph, and that any path in the "reduced" super-graph can easily be transformed to a G-discretised coordinated path.

In this example we have just performed a subdivision at 1 level of G. In the *multi-level super-graph method*, described in the next section, subdivision are performed at multiple levels. That is, subgraphs such as A and B in Figure 5 are further subdivided into smaller (connected) subgraphs. We build a hierarchy of subgraphs, with G as root, and at the lowest level subgraphs consisting of just 1 node and no edges. The idea works for more than two robots as well. Given n robots, instead of combining pairs of subgraphs into super-nodes, we will combine n-tuples of subgraphs.

6.2 Formalisation of the multi-level super-graph method

We now formalise the idea introduced in the previous section. Let $G = (V, E)$ again be a simple roadmap. A key notion in the multi-level concept is that of so called *independent G-subgraphs.* We refer to a graph $(\tilde{V}, \tilde{E})$ with $\tilde{V} \subset V$ and $\tilde{E} = E \cap (\tilde{V} \times \tilde{V})$ as a *G-subgraph.* In other words, $(\tilde{V}, \tilde{E})$ is the subgraph of G induced by $\tilde{V}$. The idea is that a set of G-subgraphs is *independent* if none of the subgraphs contains a node that blocks an edge in any of the other G-subgraphs (see also Figure 6).

Definition 7 *Let $G = (V, E)$ be a simple roadmap. Given sets $\tilde{V} \subset V$ and $\tilde{E} \subset E$, we denote $\exists x \in \tilde{V} : \exists e \in \tilde{E} : \mathcal{A}(x) \otimes \mathcal{A}(e)$ by $\tilde{V} \otimes \tilde{E}$.*
A set of G-subgraphs $\{(V_1, E_1), (V_2, E_2), \ldots, (V_n, E_n)\}$ is independent *if and only if:*

$$\forall_{i \in \{1,\ldots,n\}} : \forall_{j \in \{1,\ldots,n\}} : i \neq j \Rightarrow \neg V_i \otimes E_j$$

The basic idea of multi-level super-graphs now is that we define the super-nodes as tuples of independent and connected G-subgraphs. This makes sense, since if $\{(V_1, E_1), \ldots, (V_n, E_n)\}$ is a set of independent and connected G-subgraphs and we have $(a_1, a_2, \ldots, a_n) \in V_1 \times \ldots \times V_n$ and $(b_1, b_2, \ldots, b_n) \in V_1 \times \ldots \times V_n$, then finding a coordinated path solving the problem $((a_1, a_2, \ldots, a_n), (b_1, b_2, \ldots, b_n))$ can be accomplished by n independent graph searches in the G-subgraphs $\{(V_1, E_1), \ldots, (V_n, E_n)\}$.

For the construction of a multi-level super-graph we use a data-structure which we refer to as a *G-subdivision tree.* It gives a recursive subdivision of G into connected G-subgraphs.

Definition 8 *Given a simple roadmap $G = (V, E)$, we define a tree $\mathcal{T} = (V_\mathcal{T}, E_\mathcal{T})$ to be a G-*subdivision tree*, if it has the following properties:*

- *Each node in $V_\mathcal{T}$ is a* connected G-subgraph. *We refer to these nodes as* $\mathcal{T}$-nodes, *and to the set of $\mathcal{T}$-nodes at level*[3] *l as $V_{\mathcal{T},l}$.*
- *G is the* root *of $\mathcal{T}$.*
- *The children $(\tilde{V}_1, \tilde{E}_1), \ldots, (\tilde{V}_k, \tilde{E}_k)$ of each internal $\mathcal{T}$-node $(\tilde{V}, \tilde{E})$ are such that $\tilde{V}_1, \ldots, \tilde{V}_k$ form a* partition of $\tilde{V}$.
- *The leafs all lie at the same level, and each leaf consists of one G-node and no G-edges.*

We say that a tuple of $\mathcal{T}$-nodes $(X_1, X_2, \ldots, X_k)$ covers *a tuple of $\mathcal{T}$-nodes $(\tilde{X}_1, \tilde{X}_2, \ldots, \tilde{X}_k)$ iff $\forall_{i \in \{1,\ldots k\}}$: X_i is an ancestor of $\tilde{X}_i$. Furthermore, $(X_1, X_2, \ldots, X_k)$* covers *a tuple of G-nodes $(\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_k)$ iff $\forall_{i \in \{1,\ldots k\}} : \tilde{x}_i \in X_i$'s nodes.*

[3]We say that the root located at level 1, its children at level 2, etc.

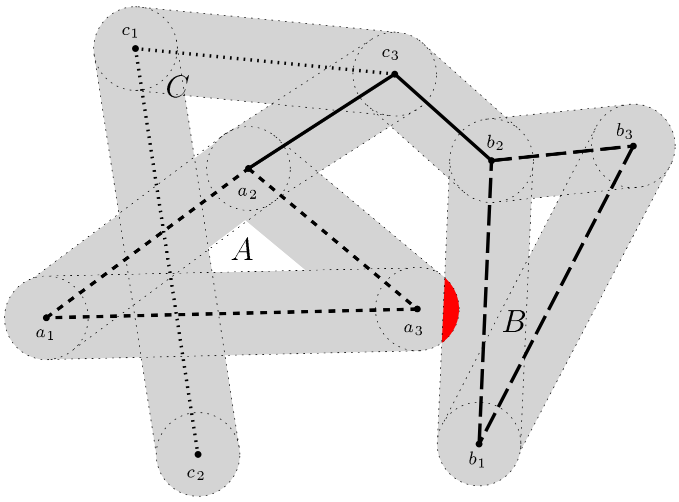


Figure 6: A simple roadmap G for a translating disc, subdivided into G-subgraphs A, B, and C induced by, respectively, $\{a_1, a_2, a_3\}$, $\{b_1, b_2, b_3\}$, and $\{c_1, c_2, c_3\}$. A and C are independent, as well as B and C. However, because the edge (b_1, b_2) is blocked by the node a_3, A and B are not independent. Hence the set $\{A, B, C\}$ is neither.

Once we have constructed a G-subdivision tree $\mathcal{T}$, we can use it to build a multi-level super-graph $\mathcal{M}^n_{G\mathcal{T}}$, which we then refer to as being $\mathcal{T}$-induced. We define the nodes of our multi-level super-graph as tuples of independent $\mathcal{T}$-nodes of the same level. The key observation with respect to the size-reduction of our super-graphs is that if a tuple of independent of $\mathcal{T}$-nodes $(w_1, \ldots, w_n)$ covers a tuple of $\mathcal{T}$-nodes $(\tilde{w}_1, \ldots, \tilde{w}_n)$, then $\{\tilde{w}_1, \ldots, \tilde{w}_n\}$ must be independent as well. This will allow us to discard potential super-nodes $(X_1, X_2, \ldots, X_n)$ that are covered by others. In this way any subdivision tree $\mathcal{T}$ defines a unique multi-level super-graph. This formalised in the following definition:

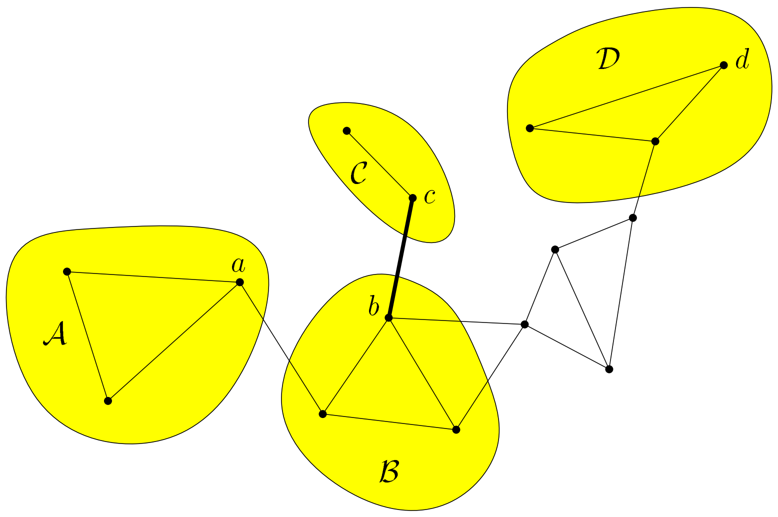
Definition 9 *Let $G = (V, E)$ be a simple roadmap, $\mathcal{T} = (V_\mathcal{T}, E_\mathcal{T})$ an associated G-subdivision tree, n the number of robots, and $\mathcal{F}^n_G = (V_\mathcal{F}, E_\mathcal{F})$ the induced flat super-graph. The* induced multi-level super-graph *$\mathcal{M}^n_{G\mathcal{T}} = (V_\mathcal{M}, E_\mathcal{M})$, is defined as follows:*

- *A tuple $(X_1, X_2, \ldots, X_n) \in V^n_{\mathcal{T},l}$ (for some l) is a node of $\mathcal{M}^n_{G\mathcal{T}}$ if and only if $(X_1, X_2, \ldots, X_n)$ is independent and no other independent $(\tilde{X}_1, \tilde{X}_2, \ldots, \tilde{X}_n) \in V_{\mathcal{T},\tilde{l}}$ covers $(X_1, X_2, \ldots, X_n)$. We refer to nodes of $\mathcal{M}^n_{G\mathcal{T}}$ as* (multi-level) super-nodes*.*

- *A pair of multi-level super-nodes (X, Y) forms an edge in $\mathcal{M}^n_{G\mathcal{T}}$ if and only if $\exists(\tilde{X}, \tilde{Y}) \in E_\mathcal{F} : X$ covers $\tilde{X} \wedge Y$ covers $\tilde{Y}$. We refer to such an edge $(\tilde{X}, \tilde{Y})$ as an* underlying $\mathcal{F}$-edge of *(X, Y) (there can be more than one). We refer to edges of $\mathcal{M}^n_{G\mathcal{T}}$ as* (multi-level) super-edges *(See also Figure 7).*

Given a simple roadmap, an associated G-subdivision tree, and the number of robots, Definition 9 defines uniquely the induced multi-level super-graph $\mathcal{M}^n_{G\mathcal{T}} = (V_\mathcal{M}, E_\mathcal{M})$. It can easily be proven that $\mathcal{M}^n_{G\mathcal{T}}$ cannot be larger than $\mathcal{F}^n_G$, but apart from this, there is no theoretical bound on the size of $\mathcal{M}^n_{G\mathcal{T}}$. So in the worst case, which could occur when simple robot is very large with respect the free space, $\mathcal{M}^n_{G\mathcal{T}}$ can be as large as $\mathcal{F}^n_G$. However, as simulation results presented in Section 8.4 indicate, in realistic cases the size of $\mathcal{M}^n_{G\mathcal{T}}$ is typically much smaller than that of the induced flat super-graph $\mathcal{F}^n_G$.[4]

[4]Also, as we will see later (Sections 6.3 and 8), it typically suffices to build only a relatively small (but properly chosen) portion of the whole multi-level structure for capturing the connectivity of the free C-space of the composite robot.



$$((a, b, d), (a, c, d)) \in E_{\mathcal{F}} \Rightarrow ((\mathcal{A}, \mathcal{B}, \mathcal{D}), (\mathcal{A}, \mathcal{C}, \mathcal{D})) \in E_{\mathcal{M}}$$

Figure 7: Assume $(\mathcal{A}, \mathcal{B}, \mathcal{D})$ and $(\mathcal{A}, \mathcal{C}, \mathcal{D})$ are super-nodes in a multi-level super-graph $\mathcal{M}^n_{G\mathcal{T}}$. If $((a, b, d), (a, c, d))$ is an edge in the corresponding flat super-graph $\mathcal{F}^3_G$, then $((\mathcal{A}, \mathcal{B}, \mathcal{D}), (\mathcal{A}, \mathcal{C}, \mathcal{D}))$ will be an edge in $\mathcal{M}^n_{G\mathcal{T}}$.

We want to stress here that the flat super-graph $\mathcal{F}^n_G$, which can be enormous for more than 3 robots, is only used for definition purposes. For the actual construction of our multi-level graph $\mathcal{M}^n_{G\mathcal{T}}$ we fortunately need not to compute $\mathcal{F}^n_G$. Given a simple roadmap $G = (V, E)$ and an associated G-subdivision tree $\mathcal{T}$, $\mathcal{M}^n_{G\mathcal{T}}$ can be computed with an output sensitive algorithm in $O\left((|V_{\mathcal{M}}| + |E_{\mathcal{M}}|) \log(|V|)\right)$ time, using $O(|V_{\mathcal{M}}| + |E_{\mathcal{M}}|)$ storage (or $O(|V_{\mathcal{M}}|)$, if one restricts $\mathcal{M}^n_{G\mathcal{T}}$ to be a forest). The computation costs for G and $\mathcal{T}$ of course depend on the used algorithms. The storage for $\mathcal{T}$ will in general be $O\left((|V| + |E|) \log |V|\right)$. We will not give details here.

The question is whether we loose any information when using the multi-level super-graph instead of the flat super-graph. This appears is not to be the case. Theorem 1 shows that multi-level super-graphs are equivalent to the flat super-graphs by which they are induced. Hence we reduce storage without losing power. Theorem 1 uses a notion of *covering* introduced in Definition 10.

Definition 10 *Let $G = (V, E)$ be a simple roadmap, n the number of robots, $\mathcal{F}^n_G = (V_{\mathcal{F}}, E_{\mathcal{F}})$ the induced flat super-graph, $\mathcal{M}^n_{G\mathcal{T}} = (V_{\mathcal{M}}, E_{\mathcal{M}})$ an induced multi-level super-graph, and $P_{\mathcal{F}}$ a path in $\mathcal{F}^n_G$ describing a G-discretised coordinated path P.*

- *A node $X \in V_{\mathcal{M}}$* covers *$P_{\mathcal{F}}$ if and only if each node x of $P_{\mathcal{F}}$ is covered by X.*
- *A path $P_{\mathcal{M}} = [X_1, \ldots, X_m]$ in $\mathcal{M}^n_{G\mathcal{T}}$* covers *$P_{\mathcal{F}}$ if and only if*

$$\exists Q_1, \ldots, Q_m : P_{\mathcal{F}} = Q_1 \oplus \ldots \oplus Q_m \wedge \forall i \in \{1, \ldots, m\} : X_i \text{ covers } Q_i$$

where $\tilde{Q} \oplus \hat{Q}$ denotes here the concatenation of the graph paths $\tilde{Q}$ and $\hat{Q}$.

- *A path $P_{\mathcal{M}} = [X_1, \ldots, X_m]$ in $\mathcal{M}^n_{G\mathcal{T}}$* covers *$P$ if and only if it covers $P_{\mathcal{F}}$.*

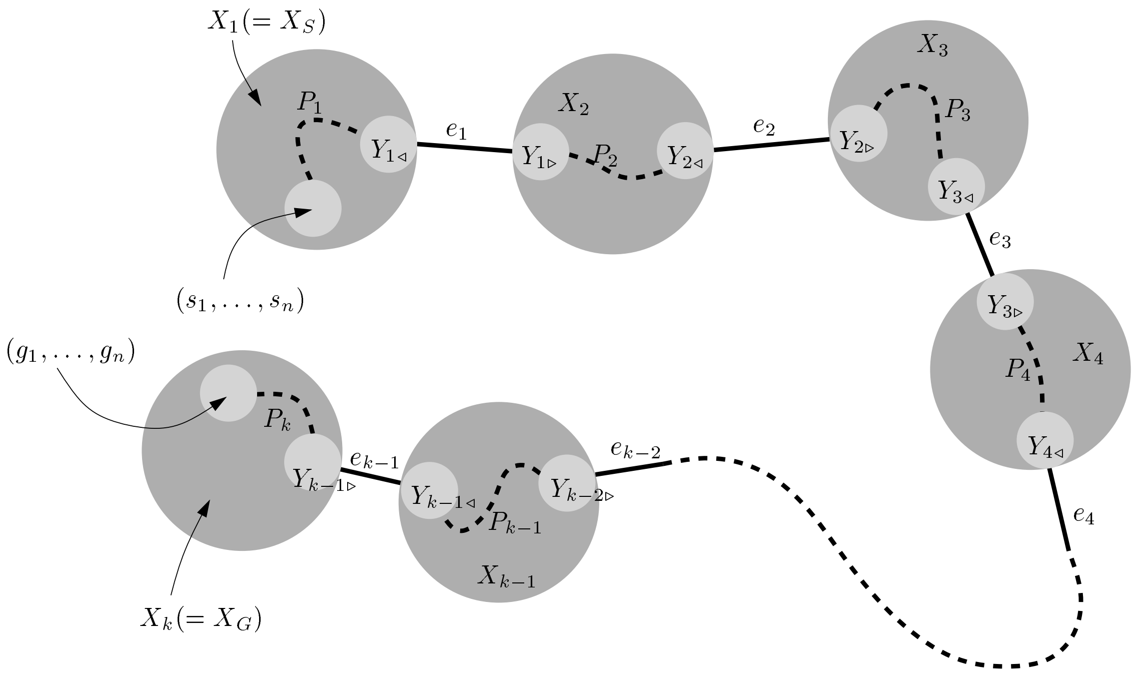


Figure 8: Transformation of a path $P_\mathcal{M} = [X_1, \ldots, X_k]$ in a multi-level super-graph to a G-discretised coordinated path $P_\mathcal{F} = P_1 \oplus T_1 \oplus P_2 \oplus T_2 \oplus \ldots \oplus P_{k-1} \oplus T_{k-1} \oplus P_k$, where T_i is the trivial coordinated path corresponding to the $\mathcal{F}$-edge e_i.

Theorem 1 *Let $G = (V, E)$ be a simple roadmap, n the number of robots, $\mathcal{F}^n_G = (V_\mathcal{F}, E_\mathcal{F})$ the induced flat super-graph, and $\mathcal{M}^n_{G\mathcal{T}} = (V_\mathcal{M}, E_\mathcal{M})$ an induced multi-level super-graph. If $P_\mathcal{F}$ is a path in $\mathcal{F}^n_G$ then there exists a path $P_\mathcal{M}$ in $\mathcal{M}^n_{G\mathcal{T}}$ such that $P_\mathcal{M}$ covers $P_\mathcal{F}$.*

Proof
Let $P_\mathcal{F}$ be a path in $\mathcal{F}^n_G$. Each $x \in V_\mathcal{F}$ is covered by exactly one $X \in V_\mathcal{M}$. This follows directly from the definition of $V_\mathcal{M}$. Hence, we can partition $P_\mathcal{F}$ into say k consecutive (and maximal) segments $Q_1, \ldots, Q_k$, such that all flat super-nodes in Q_i (for $1 \leq i \leq k$) are covered by a single multi-level super-node X_i. It remains to be shown that X_i is connected to X_{i+1} (for $1 \leq i \leq k-1$) by a multi-level super-edge. Let q_i be the last element of Q_i and q_{i+1} the first in Q_{i+1}. Because $P_\mathcal{F}$ is a path in $\mathcal{F}^n_G$ we know that $(q_i, q_{i+1}) \in E_\mathcal{F}$. Hence it follows from the definition of $E_\mathcal{M}$ that $(X_i, X_{i+1}) \in E_\mathcal{M}$. ⊔⊓

A path retrieved from a multi-level super-graph $\mathcal{M}^n_{G\mathcal{T}}$ does not directly describe a G-discretised coordinated path. Given a particular problem $((s_1, \ldots, s_n), (g_1, \ldots, g_n))$ with the s_i's and g_i's in V, one first has to map $(s_1, \ldots, s_n)$ and $(g_1, \ldots, g_n)$ to nodes X_s and X_g of $\mathcal{M}^n_{G\mathcal{T}}$, then retrieve a path connecting X_s and X_g from $\mathcal{M}^n_{G\mathcal{T}}$, and finally transform this path into a G-discretised coordinated path solving the original problem. Algorithm 2 describes how these steps can be performed (See also Figure 8).

Algorithm 2 – Multi-level super-graph path retrieval

Let $((s_1, \ldots, s_n), (g_1, \ldots, g_n))$, with the s_i's and g_i's in V, be a problem to solve.

1. *Identify nodes $X_s \in V_\mathcal{M}$ and $X_g \in V_\mathcal{M}$ such that X_s covers $(s_1, \ldots, s_n)$ and X_g covers $(g_1, \ldots, g_n)$.*

2. *Find the shortest path $P_{\mathcal{M}} = [X_1, \ldots, X_k]$ in $\mathcal{M}^n_{G\mathcal{T}}$ connecting X_s and X_g (with $X_1 = X_s$ and $X_k = X_g$). If no such path exists, then there exists no G-discretised solution to the problem (Theorem 1).*

3. *For each $i \in \{1, \ldots, k-1\}$, let $e_i = (Y_{i\lhd}, Y_{i\rhd})$ be an underlying $\mathcal{F}$-edge of (X_i, X_{i+1}), and let T_i be the trivial coordinated path corresponding to e_i.*

4. *Find a G-discretised coordinated path P_1 connecting $(s_1, \ldots, s_n)$ to $Y_{1\lhd}$, such that P_1 is* covered *by X_1.*

5. *For each $i \in \{2, \ldots, k-1\}$ find a G-discretised coordinated path P_i connecting $Y_{i-1\rhd}$ to $Y_{i\lhd}$, such that P_i is* covered *by X_i.*

6. *Find a G-discretised coordinated path P_k connecting $Y_{k-1\rhd}$ to $(g_1, \ldots, g_n)$, such that P_k is* covered *by X_k.*

7. *Let $P_{\mathcal{F}} = P_1 \oplus T_1 \oplus P_2 \oplus T_2 \oplus \ldots \oplus P_{k-1} \oplus T_{k-1} \oplus P_k$*

Now $P_{\mathcal{F}}$ is a G-discretised coordinated path solving the problem $((s_1, \ldots, s_n), (g_1, \ldots, g_n))$.

So in Step 5 we repeatedly compute a G-discretised coordinated path P_i connecting a pair of flat super-nodes $(Y_{i-1\rhd}, Y_{i\lhd})$. It is easy to see that this is always possible. $Y_{i-1\rhd}$ and $Y_{i\lhd}$ are covered by the same multi-level super-node X_i. Due to the definition of $V_{\mathcal{M}}$, we know that X_i consists of n independent and connected G-subgraphs $\tilde{G}_1, \ldots, \tilde{G}_n$. Hence, we can obtain the G-discretised coordinated path P_i by performing n independent and local graph searches in the subgraphs $\tilde{G}_1$ to $\tilde{G}_n$. An analogous argument holds for the Steps 4 and 6.

In Algorithm 2 we assume that the start and goal configurations are present as nodes in G. Algorithm 3 now describes how an arbitrary multi-robot problem can be tackled, using coordinated retractions to G and Algorithm 2.

Algorithm 3 – Multi-level super-graph method

Let G be a simple roadmap, and $\mathcal{M}^n_{G\mathcal{T}}$ an induced multi-level super-graph. Let $((s_1, \ldots, s_n), (g_1, \ldots, g_n))$ be a problem to be solved.

1. *Compute a coordinated retraction $(P_{s_1}, \ldots, P_{s_n})$ for $(s_1, \ldots, s_n)$. (If this fails, then G is of insufficient density.)*

2. *Compute a coordinated retraction $(P_{g_1}, \ldots, P_{g_n})$ for $(g_1, \ldots, g_n)$. (If this fails, then G is of insufficient density.)*

3. *Retrieve a G-discretised coordinated path $\tilde{P}$ connecting $(P_{s_1}(1), \ldots, P_{s_n}(1))$ and $(P_{g_1}(1), \ldots, P_{g_n}(1))$ (Algorithm 2). (If this fails, then there exists no G-discretised solution to the (retracted) problem.)*

4. *Let P be the concatenation of $(P_{s_1}, \ldots, P_{s_n})$, $\tilde{P}$, and $(P_{g_1}, \ldots, P_{g_n})$ reversed.*

5. *Smooth P.*

The smoothing step (5) will be treated, as mentioned earlier, in Section 7.

6.3 Sieving the multi-level super-graphs

Simulation results (see also Section 8.4) show that the size of multi-level super-graphs is considerably smaller than that of equivalent flat super-graphs. Further size-reduction can be achieved by using what we refer to as *sieved* multi-level super-graphs. From experiments we have observed that the connectivity of the free C-space of the composite robot is typically captured by only a quite small portion of $\mathcal{M}^n_{G\mathcal{T}}$, namely by that portion constructed from the relatively large subgraphs in $\mathcal{T}$. For this reason, we construct $\mathcal{M}^n_{G\mathcal{T}}$ incrementally. We sort the subgraphs in $\mathcal{T}$ by size, and pick them in reversed order of size. For each such picked subgraph we extend the super-graph $\mathcal{M}^n_{G\mathcal{T}}$ accordingly. By keeping track of the connected components in $\mathcal{M}^n_{G\mathcal{T}}$ we can determine the moment at which the free space connectivity has been captured, and at this point the super-graph construction is stopped.

This however asks for an extra action prior to the graph search in the multi-level super-graph (Step 3 in Algorithm 3). Not every flat super-node will now be covered by a multi-level super-node. Hence, given a problem (s, g) $(=((s_1, \ldots, s_n), (g_1, \ldots, g_n))$ with the s_i's and g_i's in $V)$, we must first find (G-discretised) coordinated paths connecting the flat super-nodes s and g to flat super-nodes $\tilde{s}$ and $\tilde{g}$ that are covered (by some multi-level super-nodes X and Y). We use probabilistic motions along the simple roadmap for finding such paths, as described below in Algorithm 4.

Algorithm 4 – Connecting to multi-level super-nodes

```
Let $(c_1, \ldots, c_n)$ be a flat super-node that we want to connect to a covered flat super-node.
Initially set $Q$ to be $(I(c_1), \ldots, I(c_n))$.
while $Q(1)$ not covered (by a multi-level super-node):
    Let $(c_1, \ldots, c_n) = Q(1)$.
    Pick a random $k$ from $\{1, \ldots, n\}$.
    Let $\tilde{E}$ be the set of outgoing edges of node $c_k$ (in $G$).
    Pick a random edge $e$ from $\tilde{E}$.
    Let $\tilde{Q} = (I(c_1), \ldots, I(c_{k-1}), L(e), I(c_{k+1}), \ldots, I(c_n))$.
    if $e$ not blocked by $c_1, \ldots, c_{k-1}, c_{k+1}, \ldots, c_n$
    then set $Q = Q \oplus \tilde{Q}$.
```

This algorithm fits into Algorithm 3 as follows: The probabilistic G-discretised paths, obtained by Algorithm 4, connect the 2 retraction paths (computed in the steps 1 and 2 of Algorithm 3) with the G-discretised coordinated path $\tilde{P}$ (computed in step 3 of Algorithm 3). So Step 4 of Algorithm 3) now consists of the concatenation of 5 coordinated paths.

7 Smoothing the coordinated paths

The last step in both the flat super-graph method as well as the multi-level super-graph method consists of smoothing the coordinated path found. The goal is to shortcut redundant path segments and combine alternating simple robot motions into simultaneous ones.

Again, we assume that our composite robot is a n-tuple $(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ consisting of n instances of a simple robot $\mathcal{A}$, and that $L \in \mathcal{C}^{[0,1]}$ is a local planner for $\mathcal{A}$. We use the following probabilistic algorithm for reducing the lengths of coordinated paths. Recall here that the *length* of a coordinated path is defined as the sum of the lengths of the composing simple paths.

Algorithm 5 – Smoothing a coordinated path

```
Let $P = (P_1, \ldots, P_n)$ be the coordinated path that is to be smoothed.
loop until ...
    Randomly pick $t_1, t_2 \in [0, 1]$ with $t_1 < t_2$.
    Let $(a_1, \ldots, a_n) = P(t_1)$ and $(b_1, \ldots, b_n) = P(t_2)$
    Let $Q = (L(a_1, b_1), \ldots, L(a_n, b_n))$
    if $Q$ is collision free and shorter than the $[t_1, t_2]$ segment of $P$
    then for all $t \in [t_1, t_2]$ let $P(t) = Q\left(\frac{t-t_1}{t_2-t_1}\right)$
```

So we try to replace random portions of the path P by path segments computed by the local planner for each of the simple robots. However, such a replacement takes place only if the new path segment is collision-free and shorter than the old one.

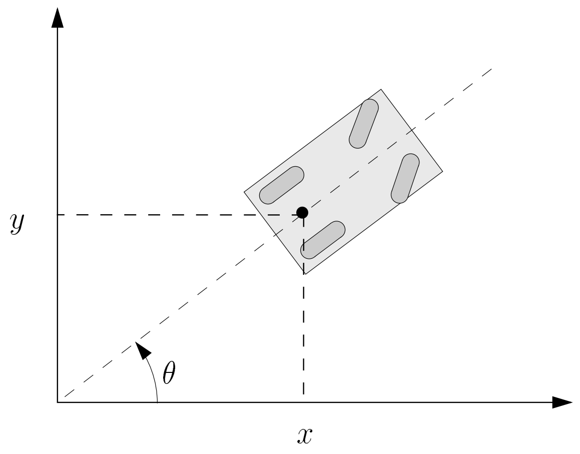


Figure 9: A car-like robot, positioned at configuration (x, y, θ).

As the dots in the above algorithm imply, the stop criterion can be chosen in various ways. A problem is that, in general, it is unknown what the length of the shortest (coordinated) path solving a particular problem is. Hence, we do not have an absolute stop criterion. Typical criteria one can use are to smooth for a fixed amount of time, or up to the point where no significant gain is achieved any longer.

Experiments with Algorithm 5 show that one obtains paths that (intuitively) look very good, in running times that are on the order of seconds.

8 Application to car-like robots and simulation results

We have applied both the flat super-graph method as well as the multi-level super-graph method to *car-like* robots. First, in Section 8.1, we describe the car-like robots that we use. Then, in Section 8.2 we briefly describe the Probabilistic Path Planner (or *PPP*) which we use for constructing the simple roadmaps. We proceed, in Section 8.3, with some details about the construction of the super-graphs (e.g., how to construct the G-subdivision trees), and finally we present simulation results for a number of problems involving three, four, and five car-like robots.

8.1 Car-like robots

A *car-like robot* is a mobile object, nonholonomically constrained in its motions such that it can only move forwards and backwards, and follow certain curves of a lower-bounded turning radius (for a formal definition, see [Lat91, ŠO95b]). Intuitively, it can perform the motions that an ordinary car can. A configuration of a car-like robot consists of three parameters x, y, and θ, where (x, y) defines the robots planar position, and θ its orientation (see also Figure 9). Hence, it has a 3-dimensional C-space.

8.2 The *Probabilistic Path Planner PPP*

The Probabilistic Path Planner (or *PPP*), which we use for the construction of the simple roadmaps, is conceptually very simple. An undirected graph $G = (V, E)$ is constructed, with nodes corresponding to free configurations of the robot and edges to simple feasible paths.

The construction is done incrementally, by repeatedly adding a *random*[5] free configuration c to V, and trying to connect c to nearby nodes in V, which we refer to as c's *neighbours*, by a *local planner*. Whenever such a connection succeeds, a corresponding edge is added to E. The local

[5]Heuristics are used for generating more nodes in certain "interesting" areas of the free C-space.

planner, which has already been introduced in Section 3, is a simple but fast planner. Given two argument configurations a and b, it constructs a path connecting a and b that is feasible in absence of obstacles. Then, it tests whether this path intersects any obstacles. If so, failure is returned, and otherwise the local planner succeeds. If the local planner is chosen properly (see also Section 9 and [ŠO95b, Šve96]), then probabilistic completeness of the (global) planner is guaranteed.

The local planner and an associated distance measure, used for selecting the neighbours of a new node, are the only robot dependent components of *PPP*. This makes the method flexible and easily applicable to different robot-types. This flexibility propagates to the multi-robot extension that we present in this paper. For more details on *PPP* we refer the reader to [OŠ94, KL94, HST94, ŠO95b, KŠLO96, Kav95].

Applying *PPP* to car-like robots asks for a local planner that computes paths that are feasible for these robots. We use the *RTR local planner*, which uses simple curves (that is, circle arcs of constant turning radii) and straight line motions for building the local paths. Given two argument configurations a and b, the planner constructs the shortest path consisting of a simple curve, followed by a straight line motion, followed by another simple curve, that connects a and b. This local planner has the properties that guarantee probabilistic completeness. For more details on *PPP* applied to car-like robots, we refer to previous work [ŠO95b].

8.3 Construction of the super-graphs

Note that, given a simple roadmap $G = (V, E)$, Definition 8 gives no unique specification of a G-subdivision tree $\mathcal{T}$. Hence, an important question is how to compute this data-structure. In our implementation we use a simple heuristic algorithm. After the root r $(=G)$ has been created, a number of its nodes are selected heuristically, and subgraphs are grown around these nodes, until all nodes of r lie in some subgraph. Heuristics are used that aim at keeping the subgraphs compact. These subgraphs form the children of r, and the procedure is applied recursively to each of these. The recursion stops at subgraphs consisting of just one node. Care is taking to build a perfectly balanced tree, by down-copying those leafs that do not lie at the deepest level of $\mathcal{T}$.

Regarding the construction of the multi-level super-graphs $\mathcal{M}^n_{G\mathcal{T}}$, we apply the “sieving” idea as described in Section 6.3. That is, we construct $\mathcal{M}^n_{G\mathcal{T}}$ incrementally, by picking the subgraphs of $\mathcal{T}$ in order of decreasing size and, for each such subgraph, extending $\mathcal{M}^n_{G\mathcal{T}}$ appropriately (as described by Definition 9). The construction is stopped at the moment that $\mathcal{M}^n_{G\mathcal{T}}$ consists of just one major component. More specifically, we stop the construction at the point where 95% of the flat super-nodes covered by nodes of $\mathcal{M}^n_{G\mathcal{T}}$ are covered by nodes of $\mathcal{M}^n_{G\mathcal{T}}$ lying in the same connected component. The computation costs for constructing a multi-level super-graph $\mathcal{M}^n_{G\mathcal{T}} = (V_\mathcal{M}, E_\mathcal{M})$, induced by a simple roadmap $G = (V, E)$, are $O\left((|V_\mathcal{M}| + |E_\mathcal{M}|)\log(|V|)\right)$, and $O(|V_\mathcal{M}| + |E_\mathcal{M}|)$ storage is used.

Finally, for solving multi-robot path planning problems with our super-graphs, we need to compute coordinated retractions (as defined in Definition 6). We do this again in a simple way. Given a tuple $(c_1, \ldots, c_n) \in \mathcal{C}_f$, for each c_i we pick the nearest node $\tilde{c}_i$ in V, and we construct $(L(c_1, \tilde{c}_1), \ldots, L(c_n, \tilde{c}_n))$. If this gives no collisions (neither between robots and obstacles and nor between robots mutually), we take this path as coordinated retraction. Otherwise the retraction fails. To remedy such failures, one must either extend the simple roadmap, or use a more powerful retraction method.

8.4 Simulation results

We have implemented both the flat super-graph method as the multi-level super-graph method in C++. We have run the resulting planners on various environments for different numbers of robots. Since the size of flat super-graphs depends exponentially on the number of robots, the flat super-graph method does not yield practical results for more than 3 robots. For this reason we only present results obtained with the multi-level planner (see [ŠO95a] for some results obtained with the flat method). The experiments were performed on a Silicon Graphics Indigo^2 workstation rated with 96.5 SPECfp92 and 90.4 SPECint92.

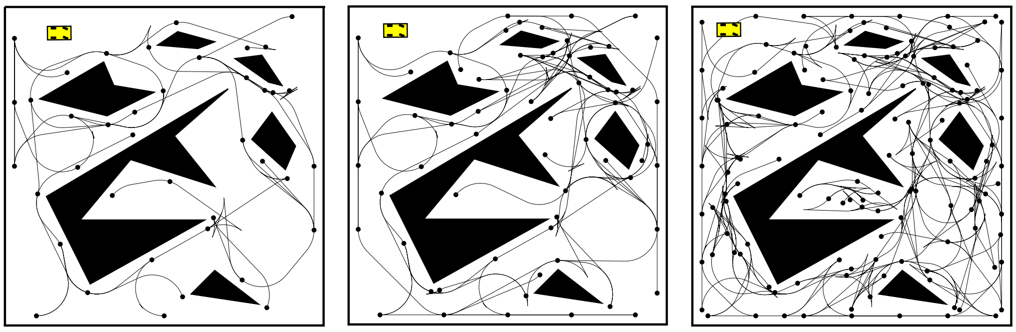


Figure 10: Scene 1, together with 3 independently constructed roadmaps. In the top-left corner of the scene we see the simple (car-like) robot.

We show results for 5 different scenes, and in each scene we present results for one or more simple roadmaps that are used for generating multi-level super-graphs. For each scene we report, for varying numbers of (simple) robots, the computations costs (in seconds) and the memory consumption of the associated multi-level super-graphs. Also we give indications of the computation costs required for retrieving coordinated paths from the super-graphs. Finally we show some snapshots of retrieved coordinated paths.

Scene 1 See Figure 10 for the first scene, and 3 independently constructed roadmaps of increasing size. The scene is relatively easy in the sense that for most path planning problems there exist many (topologically) different solutions.

In the two tables below the simulation results are given. In the left table, we see the sizes and computation costs of the simple roadmaps (1 corresponds to the leftmost roadmap, 2 to the middle roadmap, and 3 to the rightmost roadmap in Figure 10). In the right table we have, for each of the 3 simple roadmaps, the sizes and computation costs of the multi-level super-graphs for 3, 4, and 5 robots. The two leftmost columns (labelled G and n) define the used simple roadmap and the number of robots. The third and fourth column ($|V_{\mathcal{M}}|$ and $|E_{\mathcal{M}}|$) give the size of the resulting multi-level super-graph (in terms of nodes and edges). In the fifth column ($t_{\mathcal{M}}$) the computation time required for the construction of the multi-level super-graph is given (this does not include the computation time of the simple roadmap, which is given in the left table). Finally, the last column (t_r) indicates the time required for the retrieval (including smoothing) of a coordinated path from the super-graph. This value of course varies over different problems, and the values that we give are merely meant as indications.

G	$\|V\|$	$\|E\|$	t_G
1	40	58	3.4
2	64	115	7.1
3	132	274	20.2

G	n	$\|V_{\mathcal{M}}\|$	$\|E_{\mathcal{M}}\|$	$t_{\mathcal{M}}$	t_r
1	3	198	864	1.8	3.0
1	4	1008	4752	2.3	6.0
1	5	17760	103920	12.7	12.0
2	3	120	402	5.3	3.0
2	4	2184	12384	6.3	6.0
2	5	13560	69120	20.8	12.0
3	3	408	2532	30.6	3.0
3	4	2256	15216	31.4	6.0
3	5	7080	33120	47.6	12.0

So we see that the number of super-nodes lies on the order of thousands, and the computation times are less than one minute. Not surprisingly, the super-graph size increases with the number of robots. However, we see that the size of the simple roadmap does not seem to have great

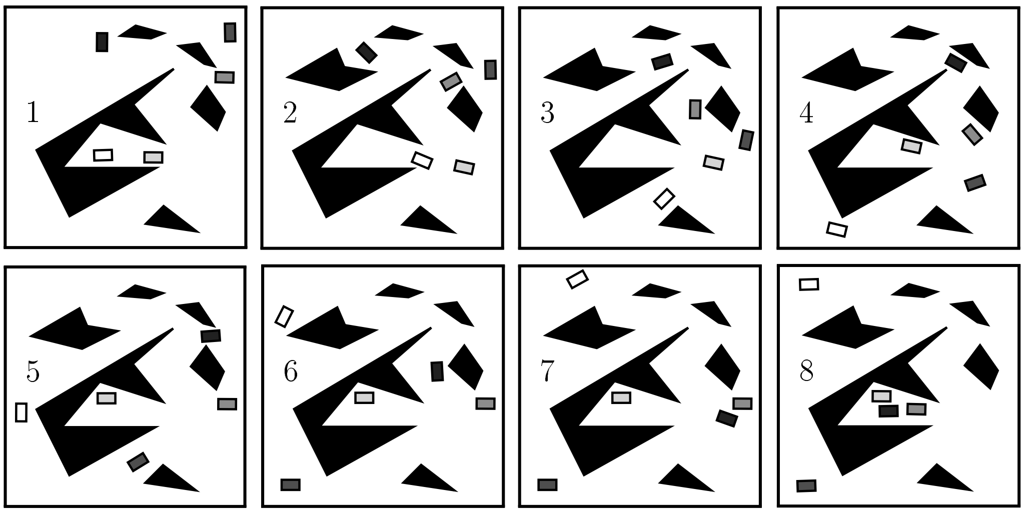


Figure 11: Snapshots of a coordinated path for 5 robots in Scene 1, retrieved from a multi-level super-graph.

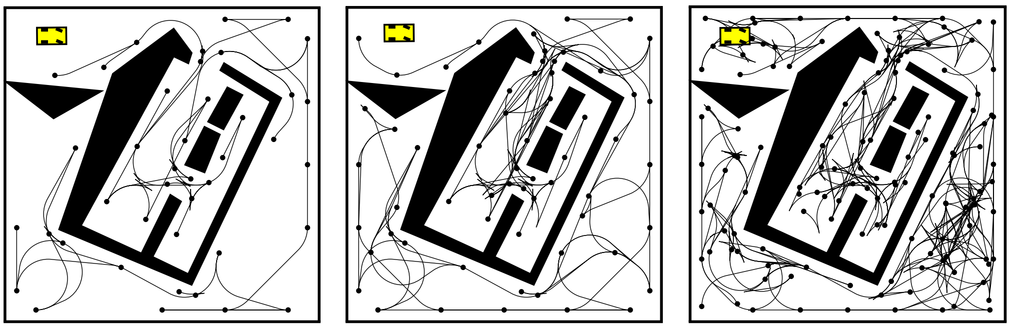


Figure 12: Scene 2.

impact on the size of the multi-level super-graph. In fact, for 5 robots we see that the super-graph size even decreases with increasing size of the simple roadmaps. This is caused by the "sieving" algorithm, as described in Section 6.3, which prunes the super-graph structure. It appears that this pruning can be performed more effectively with large (and dense) simple roadmaps than with small ones. Thanks to this, we can use simple roadmaps of high node density (such as the third simple roadmap in Figure 10).

The results given in the table say nothing about the quality of the super-graph, in terms of how well it captures the connectivity of the composite C-space, or, in other words, how many queries it does or does not solve. It is not clear how this should be measured. Experiments however make us believe that the super-graph induced by the simple roadmap 3 is sufficient for solving almost all non-pathetic problems (under which the example showed in Figure 11). This is a result of the high node density of the underlying simple roadmap.

Scene 2 See Figure 12 for the second scene, together with 3 independently constructed roadmaps. This is a more complicated scene. In particular in the middle regions, it is difficult for the robots to pass each other, and much coordination of their motions is required.

In the two tables below the simulation results are again given. These are of the same form

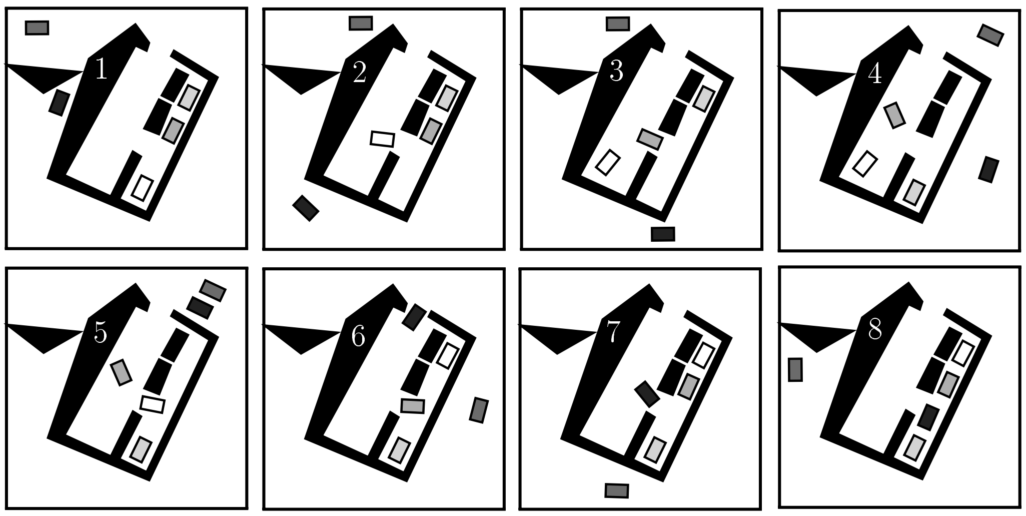


Figure 13: Snapshots of a coordinated path for 5 robots in Scene 2, retrieved from a multi-level super-graph.

as for Scene 1. We see that Scene 2 indeed requires larger super-graphs. For 5 robots and the third simple roadmap, the algorithm even failed to generate a super-graph consisting of one major component (due to memory problems). Again we see that, whereas the number of robots has large impact on the size of the data-structure, the size of the simple roadmap has not. As in the previous example, the third simple roadmap has sufficient density to yield a super-graph that solves most practical problems in the scene. See Figure 13 for snapshots of a coordinated path retrieved from it.

G	$\|V\|$	$\|E\|$	t_G
1	41	51	2.9
2	60	111	6.7
3	131	287	19.1

G	n	$\|V_{\mathcal{M}}\|$	$\|E_{\mathcal{M}}\|$	$t_{\mathcal{M}}$	t_r
1	3	186	546	3.2	5.0
1	4	7752	37968	4.6	10.0
1	5	127680	776160	132.3	30.0
2	3	732	4110	6.4	5.0
2	4	8976	52176	8.9	10.0
2	5	78120	552000	99.7	20.0
3	3	3882	22758	38.4	5.0
3	4	62520	571488	93.4	15.0
3	5	-	-	-	-

Scenes 3, 4, and 5 The Scenes 3, 4, and 5, shown in Figure 14 together with the used simple roadmaps, are of a very different nature than the first two scenes. The free space for the simple robot is very narrow, and any maneuver of one simple robot requires appropriate motions of the other robots as well. These are typical scenes where previous approaches failed.

We built multi-level super-graphs for 3 robots in Scene 3, for 4 robots in Scene 4, and for 5 robots in Scene 5. The used simple roadmaps have about 50 nodes and 120 edges, and their construction time is about 3.5 seconds.

In the following table the results for Scene 3 are given (for 3 robots). We see that in 6 seconds a multi-level super-graph is constructed that consists of 7800 nodes and almost 50.000 edges. Although the node density of the simple roadmap is quite low, most interesting problems are solved by the induced super-graph. One such problem is the so-called swapping problem, where

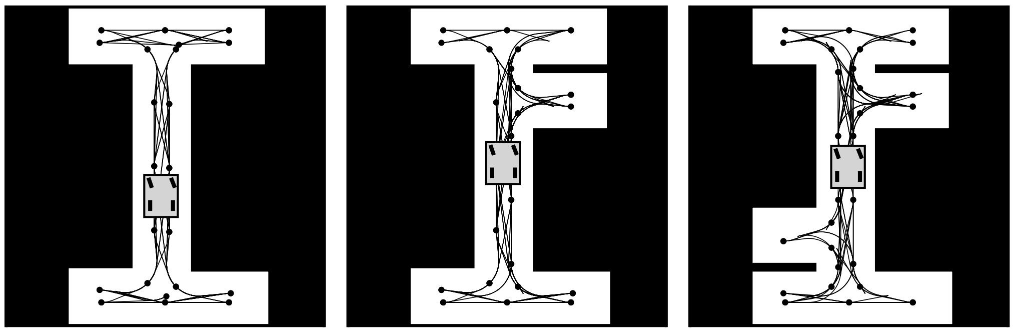


Figure 14: The Scenes 3, 4, and 5 (from left to right).

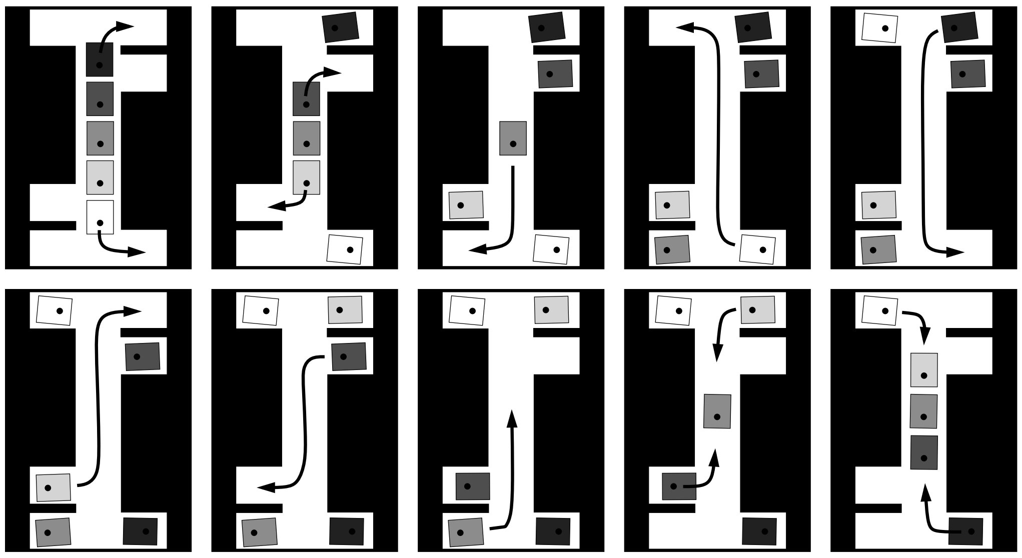


Figure 15: Snapshots of a coordinated path for 5 robots in Scene 5, retrieved from a multi-level super-graph.

the three robot are lined up in the main corridor (that is, vertically), and they are to reverse their order.

n	$\|V_{\mathcal{M}}\|$	$\|E_{\mathcal{M}}\|$	$t_{\mathcal{M}}$	t_r
3	7800	47496	6.0	1.5

For Scene 4 and 4 simple robots we have the following results. More than 40.000 super-nodes and over 250.000 super-edges are required in the super-graph for grasping the composite robots free space connectivity. The computation time is 41.1 seconds. Again the resulting super-graphs solves most interesting problems, under which the swapping problem.

n	$\|V_{\mathcal{M}}\|$	$\|E_{\mathcal{M}}\|$	$t_{\mathcal{M}}$	t_r
4	40656	265728	41.1	5.0

For 5 robots in Scene 5 things get really difficult. A super-graph with nearly 300.000 nodes and nearly 2.000.000 edges is required. The computation time is more than 20 minutes. See

Figure 15 for snapshots of a coordinated path, retrieved from the constructed super-graph, solving the swapping problem for 5 robots.

n	$\lvert V_{\mathcal{M}} \rvert$	$\lvert E_{\mathcal{M}} \rvert$	$t_{\mathcal{M}}$	t_r
5	318360	1839600	1346.7	12.0

Summarising, we can say that the super-graphs for the Scenes 3, 4, and 5 are significantly larger than those for the first two scenes. The cause for this must be that the compact structure of the free space in the Scenes 3, 4, and 5, as well as the relatively large size of the robot, cause more subgraphs to interfere. Hence, in the last three scenes, subdivision into smaller subgraphs is required.

9 Probabilistic completeness with *PPP*

In this section we show that solving G-discretised path planning problems is sufficient, in the sense that this guarantees a form of *probabilistic completeness* for *locally controllable* simple robots. We show that, if the simple roadmap is constructed properly with *PPP*, then, given a solvable multi-robot problem, the probability that there exists a G-discretised solution to this problem goes to 1 when the construction time of the simple roadmap G goes to infinity. By *proper construction* we mean that a local planner L is used that satisfies a local topological property exploiting the local controllability of the simple robot. First, in Section 9.1, we explain local controllability and we define the local topological property. Then, in Section 9.2 we give a proof of the completeness claim.

9.1 Local controllability and the local topological property

First we describe the concept local controllability (in the literature also referred to as small-time local controllability or local-local controllability), adopting the terminology introduced by Sussman [Sus83]. Given a robot $\mathcal{A}$, let $\Sigma_{\mathcal{A}}$ be its control system. That is, $\Sigma_{\mathcal{A}}$ describes the velocities that $\mathcal{A}$ can attain in C-space. For a configuration c of a robot $\mathcal{A}$, the set of configurations that $\mathcal{A}$ can reach within time T is denoted by $A_{\Sigma_{\mathcal{A}}}(\leq T, c)$. $\mathcal{A}$ is defined to be *locally controllable* iff for any configuration $c \in \mathcal{C}$ $A_{\Sigma_{\mathcal{A}}}(\leq T, c)$ contains a neighbourhood of c (or, equivalently, a ball centred at c) for all $T > 0$. It is a well-known fact that, given a configuration c, the area a locally controllable robot $\mathcal{A}$ can reach without leaving the ϵ-ball around c (for any $\epsilon > 0$) is the entire open ϵ-ball around c.

We now assume that robot $\mathcal{A}$ is locally controllable. What will be required is a local planner L for $\mathcal{A}$ that exploits the robots local controllability. This will be the case if L respects what we call the *local topological property* (*TP*), as defined in Definition 12 using the notion of *ϵ-reachability* introduced in Definition 11. We denote the ball (in C-space) of radius ϵ centred at configuration c by $B_\epsilon(c)$.

Definition 11 *Let L be a local planner for $\mathcal{A}$. Furthermore let $\epsilon > 0$ and $c \in \mathcal{C}$ be given. The* ϵ-reachable area of c by L, *denoted by $R_{L,\epsilon}(c)$, is defined by*

$$R_{L,\epsilon}(c) = \{\tilde{c} \in B_\epsilon(c) | L(c, \tilde{c}) \text{ is entirely contained in } B_\epsilon(c)\}$$

Definition 12 *Let L be a local planner for $\mathcal{A}$. We say L has the* local topological property TP *iff*

$$\forall \epsilon > 0 : \exists \delta > 0 : \forall c \in \mathcal{C} : B_\delta(c) \subset R_{L,\epsilon}(c)$$

We refer to $B_\delta(c)$ as the ϵ-reachable δ-ball *of c.*

A local planner verifying *TP*, at least in theory, always exists, due to the robots local controllability. It has been shown [ŠO95b, ŠO95c, Šve96] that a local planner for a robot $\mathcal{A}$ verifying *TP* guarantees probabilistic completeness of *PPP* for $\mathcal{A}$. In the next section we show that it also guarantees probabilistic completeness of the multi-robot super-graph planners when using simple roadmaps constructed by *PPP*.

9.2 Probabilistic completeness proof

Given a set of free configurations W and a simple roadmap $G = (V, E)$ computed by the *PPP*, we denote by $G \uplus W$ the graph $(V \cup W, E \cup \{(a,b) \in W \times V | L(a,b)$ is collision free$\})$. In other words, $G \uplus W$ is the graph that one obtains by adding the configurations in W to G together with corresponding edges, in the same way as the random nodes are added to G by *PPP*. Furthermore, we denote *PPP* using a specific local planner L by *PPP*(L). We need the notion of *C-space clearance* of coordinated paths.

Definition 13 *Let $Q = (Q_1, \ldots, Q_n)$ be a coordinated path for the composite robot $(\mathcal{A}_1, \ldots, \mathcal{A}_n)$. We say that Q has a* C-space clearance *of ϵ iff*

$$\forall t \in [0,1] : \forall (c_1, \ldots, c_n) \in B_\epsilon(Q_1(t)) \times \ldots \times B_\epsilon(Q_n(t)) : c_1, \ldots, c_n \in \mathcal{C}_f \wedge i \neq j \Rightarrow \neg \mathcal{A}(c_i) \otimes \mathcal{A}(c_j)$$

Intuitively Definition 13 states that a coordinated path Q has a C-space clearance of ϵ if anywhere in Q every simple robot can be displaced (in C-space) over a distance $\leq \epsilon$ without possibly introducing any collisions.

Theorem 2 *Let L be a local planner verifying* TP *for a locally controllable robot $\mathcal{A}$. Let $((s_1, \ldots, s_n), (g_1, \ldots, g_n))$ be an arbitrary problem for the composite robot $(\mathcal{A}_1, \ldots, \mathcal{A}_n)$, for which there exists a solution in the open free C-space of the composite robot (that is, one without robot-robot and robot-obstacle contacts). Then, within a finite amount of time,* PPP(L) *will construct a simple roadmap G such that a $\tilde{G}$-discretised solution for the problem exists, where $\tilde{G} = G \uplus \{s_1, \ldots, s_n, g_1, \ldots, g_n\}$.*

Proof
It is easily shown that the existence of a coordinated path in the open free C-space is equivalent to the existence of a coordinated path of non-zero C-space clearance, and that this again implies the existence of a discretised coordinated path of a certain positive C-space clearance.

So let P be a discretised coordinated path, solving $((s_1, \ldots, s_n), (g_1, \ldots, g_n))$, of a C-space clearance $\epsilon > 0$. Take $\delta > 0$ such that

$$\forall c \in \mathcal{C} : B_\delta \subset R_{L, \frac{3}{4}\epsilon}(c)$$

We know that such a $\delta > 0$ exists, due to the fact that L verifies the local topological property *TP*.

Since P is a discretised coordinated path, it is composed of trivial coordinated paths, that is, coordinated paths describing motions where exactly one simple robot moves. Since any subpath of a trivial coordinated path is itself a trivial coordinated path, we can subdivide P into trivial coordinated paths of arbitrarily short length. So take trivial coordinated paths $Q_1, \ldots, Q_k$ such that

$$P = Q_1 \oplus \ldots \oplus Q_k \wedge \forall i \in \{1, \ldots, k\} : \text{length}(Q_i) \leq \frac{1}{3}\delta$$

Let $c_{(i,1)}, \ldots, c_{(i,n)}, c_{(i+1,1)}, \ldots, c_{(i+1,n)}$ be configurations of $\mathcal{A}$ such that $(c_{(i,1)}, \ldots, c_{(i,n)})$ is connected to $(c_{(i+1,1)}, \ldots, c_{(i+1,n)})$ by the path Q_i (for all $1 \leq i \leq k$). Furthermore, let $m_i \in \{1, \ldots, n\}$ be such that $c_{(i,m_i)} \neq c_{(i+1,m_i)}$ (m_i is unique).

Now let $G = (V, E)$ be a simple roadmap constructed by *PPP*(L), and assume that for each $c_{(i,j)}$ there exists a node $n_{(i,j)} \in V$ within distance $\frac{1}{3}\delta$. Formally, for all $i \in \{1, \ldots, k+1\}$ and $j \in \{1, \ldots, n\}$, assume there exist $n_{(i,j)}$ such that $n_{(i,j)} \in V \cap B_{\frac{1}{3}\delta}(c_{(i,j)})$.

Let $\mathcal{F}_G^n$ be the flat super-graph induced by G (and n). Let $N_i = (n_{(i,1)}, \ldots, n_{(i,n)})$. We now claim that (1) $\forall i \in \{1, \ldots, k+1\} : N_i \in V_\mathcal{F}$, and (2) $\forall i \in \{1, \ldots, k\} : (N_i, N_{i+1}) \in E_\mathcal{F}$.

- The first claim follows directly from the definition of C-space clearance of coordinated paths. Recall that each $(c_{(i,1)}, \ldots, c_{(i,n)}) = P(t)$ for some $t \in [0,1]$, and that P has a C-space clearance of ϵ. Due to the fact that $(n_{(i,1)}, \ldots, n_{(i,n)}) \in (B_{\frac{1}{3}\delta}(c_{(i,1)}), \ldots, B_{\frac{1}{3}\delta}(c_{(i,n)}))$ and $\delta \leq \epsilon$ we know that

$$n_{(i,1)}, \ldots, n_{(i,n)} \in \mathcal{C}_f \wedge j_1 \neq j_2 \Rightarrow \neg \mathcal{A}(n_{(j_1,i)}) \otimes \mathcal{A}(n_{(j_2,i)})$$

  This means that $N_i \in V_\mathcal{F}$.

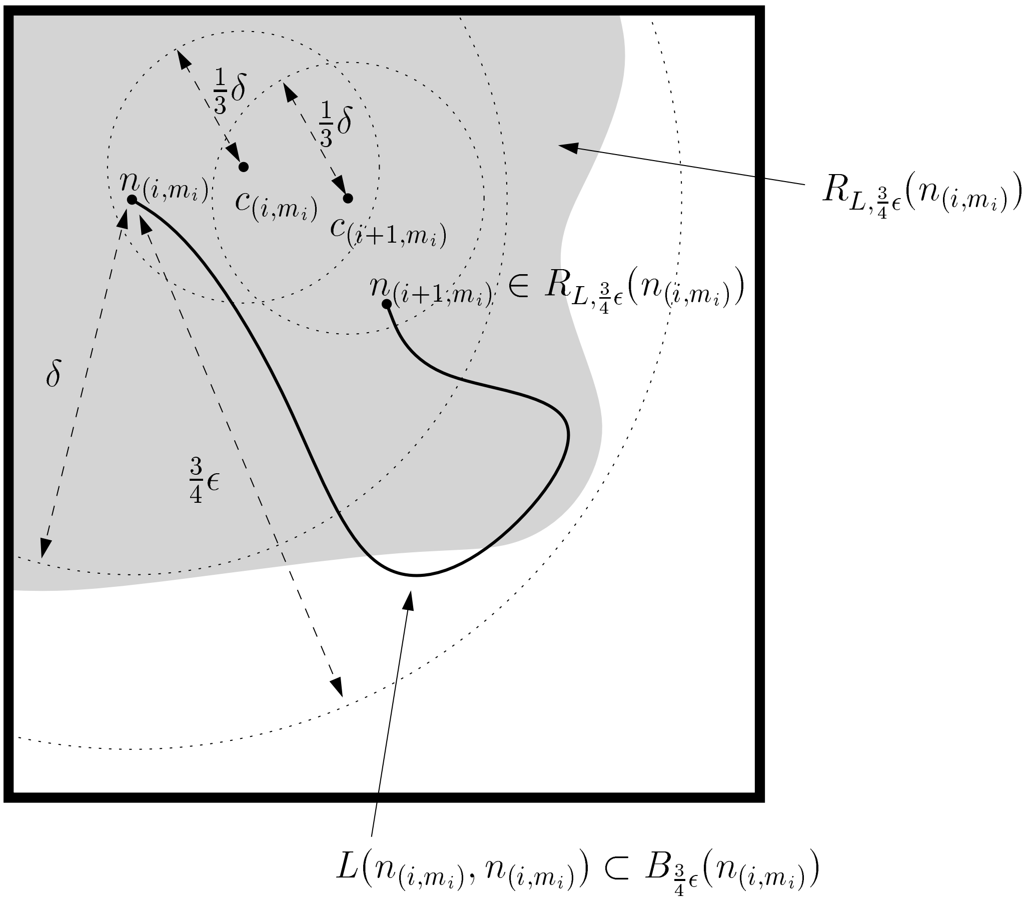

$\frac{1}{3}\delta$
$\frac{1}{3}\delta$
$n_{(i,m_i)}$
$c_{(i,m_i)}$
$c_{(i+1,m_i)}$
$R_{L,\frac{3}{4}\epsilon}(n_{(i,m_i)})$
$n_{(i+1,m_i)} \in R_{L,\frac{3}{4}\epsilon}(n_{(i,m_i)})$
δ
$\frac{3}{4}\epsilon$
$L(n_{(i,m_i)}, n_{(i,m_i)}) \subset B_{\frac{3}{4}\epsilon}(n_{(i,m_i)})$

Figure 16: Illustration to the proof of Theorem 2.

- The second claim can be proven using the local topological property of L. We have seen that $D(c_{(i,m_i)}, c_{(i+1,m_i)}) \leq \frac{1}{3}\delta$, $D(c_{(i,m_i)}, n_{(i,m_i)}) \leq \frac{1}{3}\delta$, and $D(c_{(i+1,m_i)}, n_{(i+1,m_i)}) \leq \frac{1}{3}\delta$. From these inequalities it follows that $D(n_{(i,m_i)}, n_{(i+1,m_i)}) \leq \delta$. So, due to L verifying *TP*, it follows that

  $$L(n_{(i,m_i)}, n_{(i+1,m_i)}) \text{ lies in } B_{\frac{3}{4}\epsilon}(n_{(i,m_i)})$$

  and, hence,

  $$L(n_{(i,m_i)}, n_{(i+1,m_i)}) \text{ lies in } B_{\frac{3}{4}\epsilon+\frac{1}{3}\delta}(c_{(i,m_i)}) \subset B_\epsilon(c_{(i,m_i)}).$$

  So $L(n_{(i,m_i)}, n_{(i+1,m_i)})$ lies within distance ϵ of $c_{(i,m_i)}$, and since each $n_{(i,j)}$ with $j \neq m_i$ lies within distance ϵ of $c_{(i,j)}$, it follows from the fact that $(c_{(i,1)}, \dots, c_{(i,n)})$ lies on a coordinated path of C-space clearance ϵ that

  $$\neg\mathcal{A}(n_{(i,j)}) \otimes \mathcal{A}(L(n_{(i,m_i)}, n_{(i+1,m_i)})) \text{ for } j \neq m_i.$$

  This concludes the proof of claim (2). See also Figure 16.

From (1) and (2) it directly follows that $(n_{(1,1)}, \dots, n_{(1,n)})$ is connected by a G-discretised path to $(n_{(k+1,1)}, \dots, n_{(k+1,n)})$. Furthermore, by arguments analogous to that used in the proof of claim (2), one can show that $(s_1, \dots, s_n)$ is connected to $(c_{(1,1)}, \dots, c_{(1,n)})$ and $(c_{(k+1,1)}, \dots, c_{(k+1,n)})$ to $(g_1, \dots, g_n)$ by $\tilde{G}$-discretised paths, where $\tilde{G} = G \uplus \{s_1, \dots, s_n, g_1, \dots, g_n\}$.

The assumption was that for each $c_{(i,j)}$ there exists a node in G within distance $\frac{1}{3}\delta$. Due to the random node generation of *PPP*, the probability that each (C-space) ball of radius $\frac{1}{3}\delta$ contains a node of G goes to 1 as the running time of *PPP* goes to infinity. This concludes the proof of the theorem. $\square$

Theorem 2 states that, given an arbitrary solvable problem for the composite robot, *PPP* will, within a finite amount of time, construct a simple roadmap G with which the problem can be solved. This does not say anything about the expected running times required for solving particular multi-robot path planning problems. As the results presented in the previous section illustrate, these running times are highly dependent on the complexity of the scene. Currently we are working on methods to give upper bounds on the expected running times of the algorithm, in terms of certain geometrical properties of the free C-space (see also [Šve96]).

10 Conclusions

We have presented a new approach to multi-robot path planning, that is particularly suited for constrained environments. Roadmaps for the composite robot are derived from roadmaps for the underlying simple robots. The power of the presented approach lies in the fact that only self-collision avoidance is dealt with for the composite robot, while all other (holonomic and nonholonomic) constraints are solved in the configuration spaces of the simple robots. Hence, expensive computations in the configuration space of the composite robot are avoided. Also, the method is probabilistically complete.

The roadmaps for the simple robots can be computed via *PPP*, a probabilistic single-robot method that has recently been developed and applied to a broad variety of robots. *PPP* is very time-efficient and flexible, in the sense that it is easily applied to different robots. We have shown that this flexibility propagates to the presented multi-robot extension by applying the multi-robot method to car-like robots. Simulation results indicate that the method is very efficient in both computation time and memory. Another nice property of the method is that a super-graph constructed for a particular problem can be reused for solving other multi-robot problems (in the same static environment) as well. However, the coordinated retractions, path searches in the super-graph, and smoothing, must be performed for every individual query.

There remain many possible improvements. For example, we have done only few experiments with different types of G-subdivision trees. It is important to investigate the influence of the numerous possibilities on the performance of the multi-level super-graph planner. For many applications, it even seems sensible to use characteristics of the workspace geometry for determining the subgraphs in the G-subdivision tree.

We have seen that for up to 5 independent robots the method proves practical. However, in many applications one has to deal with much larger fleets of mobile robots. Due to the enormous complexity of such systems, only decoupled planners can be used here. We think our method can be integrated into such decoupled planners for resolving deadlock situations in specific (local) workspace areas where these could arise. For example, if $\mathcal{M}$ is such an area, the global decoupled planner could enforce a rule stating that, at any time instant, no more than e.g. 4 robots are allowed to be present in $\mathcal{M}$. Path planning within $\mathcal{M}$ can then be done by a centralised planner like ours.

Acknowledgements

We thank Eric van Wessel who did some preliminary research on coordinated motion planning for his Master Thesis [vW94], and Geert-Jan Giezeman for implementing many (optimised) geometrical routines (partially contained in the “Plageo” library) used in our implementations. Finally we also thank Anil Rao, Mark de Berg, and Jules Vleugels for helpful suggestions and comments.

References

[ARIS95] R. Alami, F. Robert, F. Ingrand, and S. Suzuki. Multi-robot cooperation through incremental plan-merging. In *Proc. IEEE Internat. Conf. on Robotics and Automation*, pages 2573–2578, Nagoya, Japan, 1995.

[AS91] M.D. Ardema and J.M. Skowronski. Dynamic game applied to coordination control of two arm robotic system. In R.P. Hamalainen and H.K. Ehtamo, editors, *Differential Games - Developments in Modelling and Computation*, pages 118–130. Springer Verlag, 1991.

[BL90] J. Barraquand and J.-C. Latombe. A Monte-Carlo algorithm for path planning with many degrees of freedom. In *Proc. IEEE Intern. Conf. on Robotics and Automation*, pages 1712–1717, Cincinnati, OH, USA, 1990.

[BL91] J. Barraquand and J.-C. Latombe. Robot motion planning: A distributed representation approach. *Internat. Journal Robotics Research.*, 10(6):628–649, 1991.

[BL92] Z. Bien and J. Lee. A minimum-time trajectory planning method for two robots. *IEEE Transactions on Robotics and Automation*, 8(3):414–418, 1992.

[BLL92] J. Barraquand, B. Langlois, and J.-C. Latombe. Numerical potential field techniques for robot path planning. *IEEE Trans. on Syst., Man., and Cybern.*, 22(2):224–241, 1992.

[Buc89] S.J. Buckley. Fast motion planning for multiple moving robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 322–326, Scottsdale, Arizona, USA, 1989.

[CCL94] 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.

[CE92] H. Chu and H.A. EiMaraghy. Real-time multi-robot path planner based on a heuristic approach. In *IEEE Int. Conf on Robotics and Automation*, Nice, France, May 1992.

[CFA$^+$95] A-.H. Cai, T. Fukuda, F. Arai, T. Ueyama, and A. Sakai. Hierarchical control architecture for cellular robotic system -simulations and experiments-. In *Proc. IEEE Internat. Conf. on Robotics and Automation*, pages 1191–1196, Nagoya, Japan, 1995.

[ELP86] M. Erdmann and T. Lozano-Pérez. On multiple moving objects. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1419–1424, San Francisco, CA, USA, 1986.

[Gro88] D.D. Grossman. Traffic control of multiple robot vehicles. *IEEE Journal of Robotics and Automation*, 4(5):491–497, October 1988.

[HA92] Y. Hwang and N. Ahuja. Gross motion planning—a survey. *ACM Comput. Surv.*, 24(3):219–291, 1992.

[HSS84] J. Hopcroft, J.T. Schwartz, and M. Sharir. On the complexity of motion planning for multiple independent objects; PSPACE-hardness of the warehouseman's problem. *International Journal of Robotics Research*, 3(4):76–88, 1984.

[HST94] Th. Horsch, F. Schwarz, and H. Tolle. Motion planning for many degrees of freedom - random reflections at C-space obstacles. In *Proc. IEEE Internat. Conf. on Robotics and Automation*, pages 3318–3323, San Diego, USA, 1994.

[Kav95] L. Kavraki. *Random networks in configuration space for fast path planning.* Ph.D. thesis, Department of Computer Science, Stanford University, Stanford, California, USA, January 1995.

[KL94] L. Kavraki and J.-C. Latombe. Randomized preprocessing of configuration space for fast path planning. In *Proc. IEEE Internat. Conf. on Robotics and Automation*, pages 2138–2145, San Diego, USA, 1994.

[KNT92] S. Kato, S. Nishiyama, and J. Takeno. Coordinating mobile robots by applying traffic rules. In *IEEE IROS '92*, Raleigh, North Carolina, USA, July 1992.

[KŠLO96] L. Kavraki, P. Švestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high dimensional configuration spaces. *IEEE Trans. Robot. Autom.*, 12(4), August 1996.

[Lat91] J.-C. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Boston, USA, 1991.

[LKN$^+$89] Y.H. Liu, S. Kuroda, T. Naniwa, H. Noborio, and S. Arimoto. A practical algorithm for planning collision-free coordinated motion of multiple mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1427–1432, Scottsdale, Arizona, USA, 1989.

[LLC$^+$95] J.-H. Lee, B.H. Lee, M.H. Choi, J.D. Kim, K.-T. Joo, and H. Park. A real time traffic control scheme for a multiple AGV system. In *Proc. IEEE Internat. Conf. on Robotics and Automation*, pages 1625–1630, Nagoya, Japan, 1995.

[LT90] C.-F. Lin and W.-H. Tsai. Motion planning for multiple robots with multi-mode operations via disjunctive graphs. *Robotica*, 9:393–408, 1990.

[OLP89] P.A. O'Donnell and T. Lozano-Pérez. Deadlock-free and collision-free coordination of two robotic manipulators. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 484–489, Scottsdale, Arizona, USA, 1989.

[OŠ94] M.H. Overmars and P. Švestka. A probabilistic learning approach to motion planning. In *Algorithmic Foundations of Robotics*, pages 19–37. A. K. Peters, Boston, MA, 1994.

[SF93] T. Shibita and T. Fukuda. Coordinative behavior by genetic algorithm and fuzzy in evolutionary multi-agent system. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 760–765, Atlanta, GA, 1993.

[SG96] K. Souccar and Roderic A. Grupen. Distributed motion control for multiple robotic manipulators. In *Proc. IEEE Internat. Conf. on Robotics and Automation*, Mineapolis, USA, 1996.

[ŠO95a] P. Švestka and M.H. Overmars. Coordinated motion planning for multiple car-like robots using probabilistic roadmaps. In *Proc. IEEE Internat. Conf. on Robotics and Automation*, pages 1631–1636, Nagoya, Japan, 1995.

[ŠO95b] P. Švestka and M.H. Overmars. Motion planning for car-like robots using a probabilistic learning approach. *To appear in Intern. Journal of Rob. Research*, 1995.

[ŠO95c] P. Švestka and M.H. Overmars. Probabilistic path planning. Technical Report RUU-CS-1995-22, Dept. Comput. Sci., Utrecht Univ., Utrecht, the Netherlands, May 1995.

[ŠO96] P. Švestka and M.H. Overmars. Multi-robot path planning with super-graphs. In *Proc. CESA'96 IMACS Multiconference*, Lille, France, July 1996.

[SS83] J.T. Schwartz and M. Sharir. On the 'piano movers' problem: III. coordinating the motion of several independent bodies: The special case of circular bodies moving amidst polygonal obstacles. *International Journal of Robotics Research*, 2(3):46–75, 1983.

[SS88] M. Sharir and S. Sifrony. Coordinated motion planning for two independent robots. In *Proceedings of the Fourth ACM Symposium on Computational Geometry*, 1988.

[Sus83] H.J. Sussmann. Lie brackets, real analyticity and geometric control. In R.W. Brockett, R.S. Millman, and H.J. Sussmann, editors, *Differential Geometric Control Theory.* Birkhauser, 1983.

[Šve96] P. Švestka. On probabilistic completeness and expected complexity of probabilistic path planning. Technical Report UU-CS-96-20, Dept. Comput. Sci., Utrecht Univ., Utrecht, the Netherlands, May 1996.

[Tou86] P. Tournassoud. A strategy for obstacle avoidance and its application to multi-robot systems. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1224–1229, San Francisco, CA, USA, 1986.

[VH95] S.M. La Valle and S.A. Hutchinson. Multiple-robot motion planning under independent objectives. *To appear in IEEE Trans. Robot. Autom.*, 1995.

[vW94] E.M. van Wessel. Random motion planning voor meerdere robots. master thesis (in dutch), Dept. Comp. Science, Utrecht Univ., Utrecht, the Netherlands, June 1994.

[Wan95] J. Wang. Operating primitives supporting traffic regulation and control of mobile robots under distributed robotic systems. In *Proc. IEEE Internat. Conf. on Robotics and Automation*, pages 1613–1618, Nagoya, Japan, 1995.

[War90] C.W. Warren. Multiple robot path coordination using artificial potential fields. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 500–505, Cincinnati, OH, USA, 1990.

[WP95] J. Wang and S. Premvuti. Distributed traffic regulation and control for multiple autonomous mobile robots operating in discrete space. In *Proc. IEEE Internat. Conf. on Robotics and Automation*, pages 1619–1624, Nagoya, Japan, 1995.

[YPK79] M.Z. Yannakakis, C.H Papadimitriou, and H.T. Kung. Locking policies: Safety and freedom for deadlock. In *Proceedings of the 20th Annual Symposium on Foundations of Computer Science*, pages 286–297, 1979.