On the Completeness of Ensembles of Motion Planners for Decentralized Planning

Ross A. Knepper and Daniela Rus

***Abstract*— We provide a set of sufficient conditions to establish the completeness of an ensemble of motion planners—that is, a set of loosely-coupled motion planners that produce a unified result. The planners are assumed to divide the total planning problem across some parameter space(s), such as task space, state space, action space, or time. Robotic applications have employed ensembles of planners for decades, although the concept has not been formally unified or analyzed until now. We focus on applications in multi-robot navigation and collision avoidance. We show that individual resolution- or probabilistically-complete planners that meet certain communication criteria constitute a (respectively, resolution- or probabilistically-) complete ensemble of planners. This ensemble of planners, in turn, guarantees that the robots are free of deadlock, livelock, and starvation.**

I. Introduction

Decentralization is an important tool for enabling scalable and simple designs. Some well-intentioned efforts to decentralize motion planning and collision avoidance algorithms lead to brittleness: such algorithms may fail to find a solution when one exists—that is, they are not complete. Provable guarantees on correctness and completeness are often restrictive. In this paper, we present a set of sufficient conditions to enable a decentralized ensemble of complete motion planners to cooperatively generate a plan that is complete in the combined space across all planners. We use the term *completeness* here to refer to planners with guarantees of resolution completeness or probabilistic completeness.

In an ensemble of planners, the various sub-planners typically have distinct responsibilities that may interact with each other. For example, the planners may select motions for a single robot at different points in time, plan motions for different joints of a single robot, or plan for different robots moving in a shared space. To the extent that some fraction of the work performed by one planner does not affect the outcome of the others, efficiency can be gained by such decentralization through parallelism.

This paper contributes the following:

- the formulation of decentralized ensemble motion planning,
- sufficient conditions for the completeness of the collective action of a set of motion planners associated with individual robots in a group, and
- instantiations of the theory to multi-robot navigation tasks where complete planners that meet certain communication criteria constitute a (respectively, resolution- or probabilistically-) complete form of ensemble of planners with deadlock, livelock, and starvation guarantees.

*This work was supported by The Boeing Company.

The authors are with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, USA `{rak,rus}@csail.mit.edu`

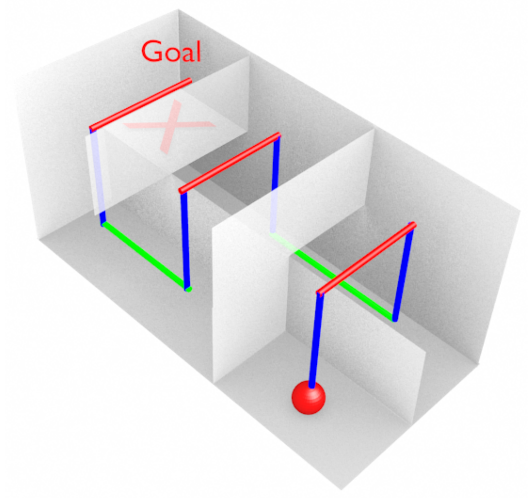


Fig. 1. An example application of an ensemble of planners. Three planners operating in orthogonal dimensions cooperate to solve a maze, with its solution color-coded according to which of three planners contributes each individual edge. The planners are X (red), Y (green), and Z (blue).

In addition, we provide several examples of decentralized planning, including a simple Cartesian grid planner, a manipulator arm, a model-based hierarchical planner, and a large team of mobile robots performing decentralized collision avoidance and navigation.

A. Related Work

Ensembles of planners have been employed (without explicit reference to the concept) for decades. An early example is Handey [20], a motion planner for a manipulator arm in which planning is decoupled to achieve computational efficiency. Handey employed separate planners for the hand and the arm/shoulder. Their operation was only loosely coupled, in that the output of the arm planner fed into the planning problem for the hand.

Around the same time as Handey, the Autonomous Land Vehicle was developed by Daily et al. [8] to accomplish outdoor autonomous navigation in off-road terrain. This ensemble of planners includes a four-level hierarchy ranging from reactive and local planning at the bottom to route and mission planning at the top. Although the levels are qualitatively different, they all represent motion in the world. Commands are fed down the chain in response to accumulated sensor input resulting from actions planned at those lower levels.

An analogous decoupling occurs in systems employing hierarchy to simultaneously solve task and motion planning problems [11, 13, 21, 22]. Given a symbolic problem statement in a geometric environment, situations can arise in which both the task and motion planning problems must be solved simultaneously to arrive at a correct solution. These two disparate planners are inherently decoupled.

In these and similar multi-planner systems, little effort was put into the general problem of ensuring that decentralized, coupled planners each achieve their goals. Instead, the designers dictated a prioritization, such that one planner effectively becomes the slave of another. In this paper, we examine that general question of how decentralized, coupled planners can achieve completeness guarantees without prioritization schemes.

The subject of completeness has a long history within the robotic motion planning community. A necessary condition of completeness is the configuration space, first described by Lozano-Perez [19]. The Cspace enables planners to uniquely and precisely describe every possible configuration of a robot. Soon thereafter, Brooks and Lozano-Perez [4] described a system in which a regular, discrete sampling of states within the Cspace enabled a motion planner to exhaustively search for a collision-free path in a grid at a fixed resolution. The guarantee of finding an existing path with such a search would later be termed *resolution completeness*.

Canny [5] shows that complete motion planners require time that is exponential in the dimension of the search space, thus motivating the desire for decoupling of dimensions where possible.

The notion of *probabilistic completeness* was introduced by Barraquand and Latombe [2]. This guarantee provides only that the probability of finding an existing path asymptotically approaches one. Such probabilistic planners still obey the complexity bounds given by Canny—however they provide two advantages over grid-based search. First, they incrementally and uniformly refine sampling resolution throughout the search space. Second, they solve certain classes of "easy" problems quickly. In expansive spaces [12], straight-line paths may cover large distances efficiently.

Comparable performance to a probabilistic planner may be obtained along with the stronger resolution-completeness guarantee by the use of deterministic sampling sequences, as described by LaValle et al. [17].

II. Ensemble Motion Planning Formulation

In this section, we describe and formalize a common scenario in robotics, wherein several motion planners operate in parallel on different aspects of a motion planning problem. Each individual planner accepts its own input and returns its own output. Communications among the planners connect their inputs and outputs in a manner that is carefully constructed to produce a correct and complete global result. In effect, the ensemble of motion planners appears as though it were a single, centralized planner, although it may realize certain benefits from distributing its computation.

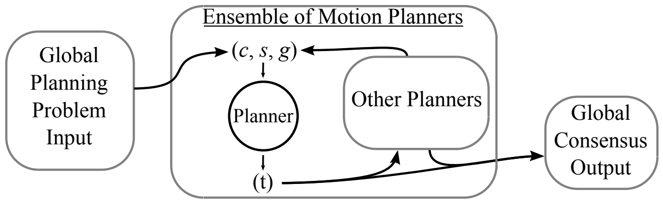


Fig. 2. The components of a communication graph. Two or more self-contained motion planners comprise the ensemble. Each takes inputs first from the global planning problem specification, followed by the other planners. Inputs are **constraints**, **start** state, and **goal** state set. Output, a **trajectory**, informs other planners. The planners form consensus around a global plan that solves the global problem.

Figure 2 illustrates motion planners at two scales. Let a *motion planner* be a function

$$\mathbb{P}_i \colon \mathbb{C} \times \mathbb{R}^{n_i} \times \mathbb{R}^{n_i} \to \mathcal{P}. \tag{1}$$

In the figure, the planner inputs are abbreviated (c, s, g). Here, $c \subset \mathbb{C}$ represents the constraint set; constraints include obstacles, kinodynamic constraints, and anything else that restricts how a robot moves instantaneously from a given configuration. The two parameters $\{s, g\} \subset \mathbb{R}^{n_i}$ represent the start and goal state, where n_i is the dimension of the state space for planner $\mathbb{P}_i$. The planner output, $t \in \mathcal{P}$, represents a path or trajectory of the form $t \colon [0, t_f] \to \mathbb{R}^{n_i}$, with t_f the length of time for which the trajectory executes.

In this formulation, the individual planners always return a path. If no solution is discovered, the planner will return a path that makes some motion. The planner attempts to make progress towards the goal, but it is impossible to know whether a partial plan actually makes progress. Therefore, if the planner is repeatedly presented with the same input, it should produce a different output every time to help explore the full space. Cooperating planners may then be able to improve upon these partial solutions.

A. Scope

Implicit inputs to a motion planner, not shown in Fig. 2, describe the *scope* of the problem. The scope defines the tools at the planner's disposal to solve a planning problem. Scope for planner $\mathbb{P}_i$ is composed of a tuple

$$\mathbb{S}_i = (Q_i, T_i, U_i), \tag{2}$$

which is interpreted as follows.

- Q_i – state space variables (i.e. $\mathbb{R}^{n_i}$)
- T_i – time
- $U_i(q_i, t_i)$ – action set as a function of state and time

Each of Q, T, and U are subject to discretization or sampling for the purpose of conducting search. A planner produces a trajectory—that is, a series of motions within the state space—by exploiting the full domain of each scope parameter.

B. Ensemble Motion Planning

Each individual planner in the ensemble of motion planners shown in Fig. 2 attempts to contribute a portion of the solution to a larger problem. A *motion planning problem* comprises a tuple

$$\mathcal{M}_i = (\mathbb{P}_i, \mathbb{S}_i, c_i, s_i, g_i), \tag{3}$$

in which $\mathbb{P}_i$ indicates a motion planner and $\mathbb{S}_i$ specifies the scope in which the problem must be solved. The remaining parameters represent inputs to the planner. As before, c_i is a set of constraints; $s_i \in Q_i$ and $g_i \in Q_i$ are the start and goal states.

We now establish the definition of an ensemble of planners. Two separate planning problems $\mathcal{M}_i$ and $\mathcal{M}_j$ form an *ensemble of planning problems* $\mathcal{M}_{ij}$. They are solved by an *ensemble of planners* $\mathbb{P}_{ij}$ operating on the joint state space,

$$Q_{ij} = Q_i \times Q_j, \tag{4}$$

subject to constraints

$$c_{ij} = c_i \cup c_j \cup c_{coupled}(Q_i, Q_j). \tag{5}$$

The term $c_{coupled}(Q_i, Q_j)$ represents a set of constraints derived from the fact that certain dimensions may be coupled between Q_i and Q_j.

Coupling between state variables occurs when the planners operate in partially overlapping spaces. For example, an ensemble of two planners plans in Cartesian XYZ space, such that the first planner controls XY and the second controls YZ. The Y dimension is coupled between the two planners, bringing the number of freedoms down to the expected three. We return to this example shortly.

To be clear, the combination of constraints between two planners can never result in an overall reduction of total constraints. Although the combination of scope may open up new parts of the space for access by the robot that were previously inaccessible due to an obstacle or a nonholonomic constraint, those constraints persist and prevent the same motions from the same configurations as before.

C. Communication Graph

Returning to Fig. 2, the connections joining inputs and outputs of each planner within an ensemble form a directed communication graph. In addition, the graph includes global inputs and outputs to the ensemble of planners.

Completeness places several demands upon the structure and usage of the communication graph, The most common means of inter-planner communication is to connect one planner's output trajectory to another planner's input parameters—one of **constraints**, **start**, or **goal**.

An alternative style of communication occurs when planners run at significantly different replan rates, one planner may effectively invoke another planner, taking its results as an internal parameter. We present such an example in Section III-A, Hierarchical Planner.

There also exists a global input to all planners, in the form of the original problem specification. Thus, any inputs not provided by other planners default to these global values. Naturally, there is also a global output in the form of a consensus solution to the planning problem.

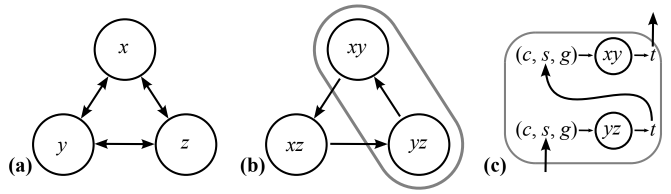


Fig. 3. Two example communication graphs for a Cartesian grid planner. **(a)** Each of the X, Y, and Z dimensions has its own dedicated planner and the communication graph is fully connected (all plans are shared directly with all planners). This scenario corresponds to the trajectory (t) shown in Fig. 1. **(b)** Three planners exhibit some redundancy, planning in the XY-, XZ- and YZ-planes respectively. Although the graph is not fully connected, progress made by any planner can still be shared indirectly with all others. **(c)** The end-state of an output trajectory becomes the start state for another planner.

For planners that operate in a subspace of Q_i of lower dimension, it should be noted that uncontrolled state variables pass through unchanged to any receiving planner. State variables not controlled by the planner take the initial value of the global input and are later assigned the value of those parameters passed in. We provide a simple example of this concept in Section II-E.1, Cartesian Grid Planner.

D. Division of Scope

Next we consider several methods of dividing work among planners. Each approach leads to a particular choice of interface between an output trajectory and input parameters.

a) Time: Planners may plan for common state variables, or subsets thereof, at different time intervals. In this case, a state from one planner's trajectory typically defines another's start or goal state.

b) Space: Planners may occupy distinct dimensions or regions of the state space over a common time interval. One planner's trajectory alters another's constraints. This method of coordination may involve a collection of robots, in which each planner's output trajectory comprises a dynamic obstacle to the other planners. Alternatively, several planners might control different joints of a common kinematic linkage.

c) Action: Planners may explore separate portions of the overall action set. In such an arrangement, planners may overlap in both time and space.

d) Disjoint: Two planners may be completely independent of one another, in which case they partition the global scope. If a planning problem can be decomposed into disjoint subproblems, then an ensemble of complete planners is trivially complete, and solutions are no harder to find than the most difficult constituent planning problem. In this case alone, the communication graph may be disconnected because there is no need for consensus.

E. Examples

To help fix these concepts, we present two simple examples demonstrating ensembles of planners.

Algorithm 1 Dijkstra(G, s, g)

```
Input: $G$ ▷ a connected graph to search
Input: $s$ ▷ the start state
Input: $g$ ▷ the goal state set
 1: for all vertices $v$ in $G$ do
 2:     $d[v] \leftarrow \infty$
 3:     $p[v] \leftarrow$ undefined
 4: Let $PQ$ be a priority queue          ▷ states sorted by cost
 5: $PQ$.insert( $(s, 0)$ )                          ▷ tuple
 6: $d[s] \leftarrow 0$
 7: while $Q \neq \emptyset$ do
 8:     $(u, d) = PQ$.pop()
 9:     if $d = \infty$ then
10:         return None
11:     if $u = g$ then
12:         return Trace_Back($g, s, p$)   ▷ sequence of nodes
13:     $r \leftarrow \emptyset$                        ▷ recent update list
14:     for all $v$ in $u$.outbound_node_neighbors() do
15:         $c \leftarrow d +$ dist_between($u, v$)
16:         if $c < d[v]$ then
17:             $d[v] \leftarrow c$
18:             $p[v] \leftarrow u$
19:             $PQ$.replace( $(v, c)$ )         ▷ replace same state
20:             $r$.append( $(v, c, u)$ )   ▷ tuple: (state, cost, prev)
21:     Exchange_Data($r$, $d$, $p$, $PQ$)
22: return None
```

Algorithm 2 Exchange_Data(r, d, p, PQ)

```
Input: $r$ ▷ recent updates
Input: $d$ ▷ table of shortest inter-node distances
Input: $p$ ▷ table of actions to reach states
Input: $PQ$ ▷ priority queue
1: for all $n$ in outbound_planner_neighbors() do
2:     send($n, r$)
3: for all $n$ in inbound_planner_neighbors() do
4:     $i$ = receive($n$)                         ▷ inbound list
5:     for all $(v, c, u)$ in $i$ do    ▷ tuple: (node, cost, prev)
6:         if $c < d[v]$ then
7:             $d[v] = c$
8:             $p[v] = (u, n)$   ▷ augmented with source planner
9:             $PQ$.replace( $(v, c)$ )          ▷ replace same state
```

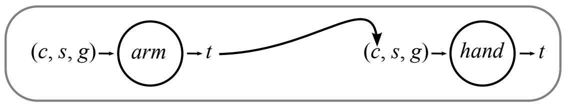


Fig. 4. The communication graph for Handey. The arm planner is completely decoupled from the hand planner, thus the graph is not strongly connected.

1) Cartesian Grid Planner: Suppose a set of planners search over subsets of the full 3D Cartesian state space, XYZ. The planners cooperate to find a path through a maze, as in Fig. 1. Scope is divided among the planners by state space dimension and action, as shown in the example ensemble communication graphs in Fig. 3. No single planner is capable of solving the maze alone, so they must exchange partial plans. The terminus of one planner's partial trajectory feeds into another planner's start state. Some efficiencies are possible, as each planner only needs to have knowledge of the subset of obstacles blocking motion along certain dimensions.

As the planners operate in parallel, each offers partial solutions for the others to build upon. The basic Dijkstra's algorithm [10] can be extended by augmenting each state's back pointer with the identity of the planner that reached that state, so that the whole ensemble of planners can reconstruct the path leading to the goal. Alg. 1 gives the basic Dijkstra's search. The only change is Line 21, which adapts the algorithm for an ensemble of planners. The implementation of Exchange_Data() is given in Alg. 2.

2) Arm Planner: Similar to the Cartesian planner, we can decompose a robot arm into the separate joints or groups of joints. Precedent for such a decoupled arm planning approach goes back at least to Handey [20], which planned for the arm's shoulder and elbow separately from the hand. By separating out the hand planning, Handey's planning problem remained efficiently computable for computers of the day. In order to decouple the motion of the hand, a large enough bounding box was maintained around the hand when collision checking to accommodate all possible hand orientations. As presented, Handey's planning ensemble actually represents two disjoint planning problems, although the arm planner's output places a kinematic constraint on the hand planner. This decoupling is practical only because the total swept volume of all possible hand motions encompasses a relatively small fraction of the total reachable volume of the arm.

One can imagine a modification to Handey in which the two planners do communicate bilaterally. The arm and hand planners might each assume a particular trajectory for the other, iteratively exchanging plans until two mutually agreeable trajectories can be achieved. Such an approach would enable Handey to reach into tight spaces prohibited by the bounding box.

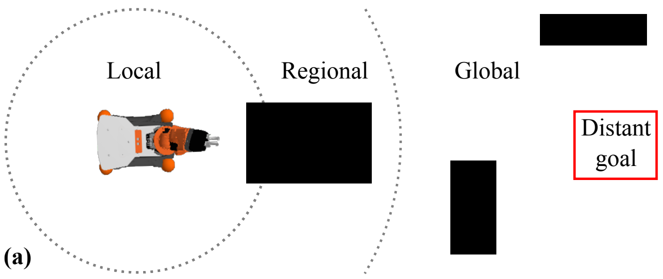
III. Applications

Now we move on to some more complex applications of the ensemble of planners.

A. Hierarchical Planner

In a hierarchical planner, responsibility for planning parts of a path to the goal is divided spatially, as in Fig. 5, or in time. The motivation for such a division is to make motion planning tractable at large scales. At short range, kinodynamic constraints are important, but the combination of such constraints with obstacle avoidance scales poorly. Thus, planners at larger scales trade lower fidelity for increased scalability.

Planners at different scales have different replan rates. The local planner replans often to stay reactive. Larger-scale planners replan less frequently and return responses more



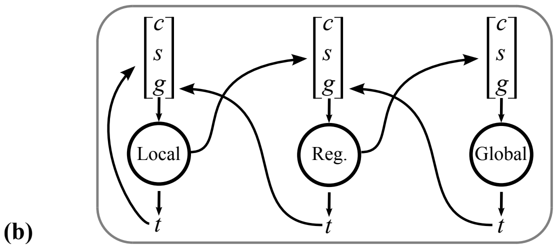


Fig. 5. Communication graph for a hierarchical planner. **(a)** Planning authority is divided in time between the immediate future (local planner), intermediate future (regional planner), and the more distant future (global planner). **(b)** The planners run at different rates. Larger scale planners employ less fidelity, and their plans therefore are quicker to compute and change less often, enabling greater reuse. Each pair of neighboring planners includes two communication cycles. Within a single local replan iteration, the global planner is invoked repeatedly to provide many completions of the local plan that begin where it ends. Then, as each local replan cycle terminates, the robot executes a part of the output trajectory. The local planner then begins planning from a new start state.

quickly to queries based on the most recent replan results, thus providing scalability.

The model-based hierarchical planner (MBHP) [15] is one example of a hierarchical planner with two levels – local and global. This planner utilizes action-space sampling at the local level, bounded by a fixed-time planning horizon. The planner integrates sampled actions through a high-fidelity motion model to compute output trajectories. At the global level, D* [16] quickly computes a path from the endpoint of each local trajectory in a discretized square grid search space. With bounded robot velocity, all queries to the global map occur in spatial proximity and so D* leverages previous results in responding to local planner queries. The ensemble of planners selects the combined local+global path with minimum length.

B. Decentralized Collision Avoidance

Multiple robots frequently operate in a shared task space and must avoid collision with each other as well as with static obstacles. We address a particular mobile robot collision avoidance algorithm [14] inspired by human navigation.

In this paradigm, each mobile robot runs its own instance of MBHP. Let us assume that each robot communicates with another only if their local planners generate interfering paths, which they can determine based on observed proximity. Each robot communicates bidirectionally with those neighbors within range of its local planner. Robots may be spread out over a wide space, such that no single robot can directly communicate with all others (Fig. 6). However, each robot

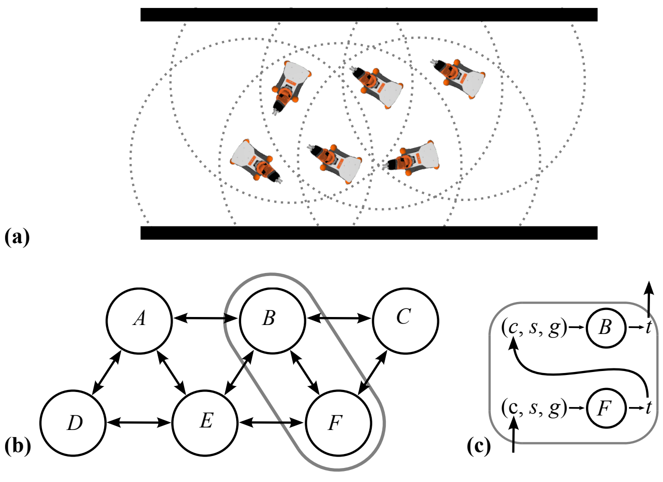


Fig. 6. Example communication graph for distributed collision avoidance. **(a)** A possible configuration of robots with communication radii shown. Robots can directly exchange plans only with others inside of the communication radius. **(b)** The communication graph corresponding to the above configuration. Encircled nodes are shown zoomed. **(c)** An output trajectory informs constraints of neighboring planners by defining a dynamic obstacle.

only needs to reach consensus on its own state variables before proceeding to execute its own planned trajectory. Planners exchange local trajectories output by MBHP until a mutually agreeable set of trajectories with one's neighbors has been discovered.

In selecting trajectories, it is difficult to converge in a single step to a set of non-colliding plans. In order to allow the robots to make forward progress while continuing to search, the initial threshold allows up to 50% overlap with other robots' planned trajectories. We say that two robots' trajectories overlap only if they result in the robots occupying some space at the same time. The concession to permit overlap recognizes the need for robots to incorporate one another's response to their own presence. In particular, all robots can be expected to make equal effort to avoid a collision. Thus, if both robots avoid each other 50%, then the robots smoothly avoid one another entirely [23]. However, this algorithm alone does not result in a complete motion planner.

Completeness requires sufficient planning time to conduct a full search. If the robots must plan while following an earlier partial plan, then there is no guarantee that multiple robots will find a joint solution before they collide with each other. Consequently, we require that the robots must possess a contingency plan, as described by Bekris et al. [3].

In the contingency plan framework, the planner actually produces two trajectories each cycle. The first is a conventional proposed path to reach the goal, whereas the second is a contingency plan that has no end time—it can continue to be executed as long as necessary for the ensemble of planners to converge to a solution. For mobile robots, this contingency trajectory typically brings the robot to a halt. The main plan can be violated by other robots until the final plan is found, but the contingency plan cannot. Each robot must absolutely avoid another's contingency trajectory, executing its own contingency plan if necessary to do so. The ability to "park" the robots for arbitrarily long enables them to execute a complete planning algorithm.

Additionally, some memory is needed to ensure completeness, which entails a minor modification to MBHP. Specifically, each robot's local planner must remember what actions it has proposed in the past in response to the proposals of other robots. If the set of a robot's neighbors proposes the same set of actions as in a previous replan step, then the robot must propose a different action than it has proposed before. This policy guarantees that the joint state space of all the robots will be explored, thus ensuring resolution completeness.

IV. Completeness

We prove a set of sufficient conditions for the completeness of an ensemble of motion planners. The basic completeness notion requires that a motion planner will always return a solution path if one exists within the scope of the motion planning problem. We consider two limited forms of completeness.

A. Resolution Completeness

A *resolution-complete* motion planner is guaranteed, in finite time, to find a collision-free path to the goal whenever one exists (up to the resolution of the discretization), and otherwise to return failure when there is no solution at that resolution [2]. Discretization applies to the attributes of state space, action space, and time.

If a given choice of discretization results in a failure to find a solution to a motion planning problem, then the algorithm can try again with a finer discretization in one or more attributes. Repeated refinement of the resolutions of discretization leads to a sampling that is dense in the limit.

B. Probabilistic Completeness

If a planning problem has a solution, then a *probabilistically complete* motion planner's probability of failing to find a solution approaches zero as time goes to infinity. If no solution exists, then the planner may either run forever or terminate after some iteration count with the response that the existence of a solution is unknown.

Probabilistically complete planners often sample from a random distribution that is dense in the underlying space. Note that a resolution-complete planner that automatically refines its sampling resolution is also probabilistically complete.

C. Conditions for Completeness

We are specifically concerned here with demonstrating the completeness of an ensemble of planners, given the corresponding completeness of the constituent planners. We therefore briefly lay out several conditions on how the ensemble of planners is connected together and how it conducts search of the joint state space.

In general, a complete ensemble E need not comprise the largest possible connected graph of planners, G. Instead, we may select any subgraph $E \subset G$ and scope $\mathbb{S}_E$ consistent with a planning problem $\mathcal{M}_E$ over that scope such that the following criteria are met.

1) Monopoly on State Variables: Consensus requires that E must have a monopoly within G on the state variables of Q_E. A monopoly implies that no planner outside of E

- have control over any state variable within Q_E, nor
- have influence over the inputs of any planner with control over some state variable in Q_E.

2) Strongly Connected Communication Graph: In addition to monopoly, consensus algorithms place requirements on the connectivity of the communication graph. Planners in E that plan in $\mathbb{S}_E$ must be strongly connected. A graph E is *strongly connected* if for every pair of nodes p and q in E, p is reachable from q and q is reachable from p [1]. Any remaining planners in E do not plan in $\mathbb{S}_E$ but influence the inputs of planners that do. Such planners must be at least weakly connected (i.e. unidirectionally). They are not required to receive inputs back because they do not participate in the consensus algorithm.

The particular protocol for establishing consensus is not the focus of this paper. An established consensus protocol such as Paxos [6] might be employed for this task.

3) Scope Coverage: Completeness requires that the union over each element of $\mathbb{S}_E$ (that is state dimensions, time, and actions) must cover the entire search space in each respective domain of the ensemble planning problem. In general, this requirement means that the ensemble of planners must search over all dimensions in Q_E for all time $T_E = [0, \infty)$ and all valid actions U_E of the robot.

4) Countable Combinatorics: We consider motion planning as a search over a tree of actions parametrized by time. LaValle [18, pp 691-2] gives two properties that are sufficient to show resolution completeness:

- the same motion primitive is never applied twice from the same vertex, and
- the planner varies discretization resolution as needed.

Here, a motion primitive comprises some finite-length sequence of actions available from a given state.

5) Probabilistic Combinatorics: For probabilistic completeness, a looser guarantee is sufficient. We require only that there is some positive probability of taking any feasible action from any state. This requirement imposes an independence condition on all of the planners' samplers. Note that the countable combinatorics listed above satisfy these criteria, and so every resolution-complete planner is inherently probabilistically complete as well.

D. Completeness Theorems

Lemma 1: Assume an ensemble of motion planners $\mathbb{P}_E$ with scope $\mathbb{S}_E$ and communication graph E, whose connectivity remains fixed over the length of motion planning process, and that each constituent planner $\mathbb{P}_i \in \mathbb{P}_E$ is provided with a common global specification of the planning problem. Given conditions 1 and 2, $\mathbb{P}_E$ is capable of producing any output trajectory that an equivalent centralized planner with scope $\mathbb{S}_E$ could generate. ■

Proof: In order to behave as a centralized planner, $\mathbb{P}_E$ must be capable of discovering and forming consensus on any possible trajectory over the full scope of the ensemble, $\mathbb{S}_E$. Each planner $\mathbb{P}_i$ must be capable of proposing any plan within $\mathbb{S}_E$, by a combination of internal search and external communication. To the extent that $\mathbb{P}_i$ does not internally plan over $\mathbb{S}_E$, it must receive partial plans covering the complement of its own internal scope, $\mathbb{S}_c = \mathbb{S}_E \setminus \mathbb{S}_i$. Strong connectedness guarantees that any partial plan can be relayed to where it is needed in the graph, and scope coverage guarantees that such a partial plan is available through some combination of other planners in the graph. Forming consensus requires strong connectedness and monopoly. ■

Theorem 1: Let $\mathbb{P}_E$ be an ensemble of resolution-complete planners satisfying the conditions on monopoly, strong connectedness, scope coverage, and countable combinatorics outlined in Section IV-C. The motion planner $\mathbb{P}_E$ is resolution-complete. ■

Proof: A centralized motion planner with countable combinatorics is resolution complete by definition. By Lemma 1, a decentralized ensemble of resolution complete planners behaves like a centralized planner. The individual planners, being resolution complete, never apply a single motion primitive more than once. Since they are aware of the full scope, they can apply this guarantee to the full scope as well, thus never repeating actions from the same full state. Since each planner can individually vary discretization as needed, the ensemble can do so as well. Therefore, the ensemble of planners is resolution complete. ■

Theorem 2: Let $\mathbb{P}_e$ be an ensemble of probabilistically-complete planners satisfying conditions on monopoly, strong connectedness, scope coverage, and probabilistic combinatorics outlined in Section IV-C. $\mathbb{P}_e$ is probabilistically complete. ■

Proof: A centralized motion planner with probabilistic combinatorics is probabilistically complete by definition. By Lemma 1, a decentralized ensemble of probabilistically-complete planners behaves like a centralized planner. The individual planners, being probabilistically complete, apply every motion primitive with non-zero probability. They can apply this guarantee to the full scope as well, thus sampling all actions from the same full state with non-zero probability. ■

Next, we prove that MBHP is resolution complete for a class of robot called small-space controllable (SSC) [9]. Small-space controllability stipulates that for any configuration q and radius $\epsilon > 0$, there exists a neighborhood $N(q)$ around q such that for all configurations $r \in N(q)$ there exists an admissible trajectory from q to r that is contained within an ϵ-ball centered at q. Such a robot is capable of following a given trajectory with arbitrarily small error if permitted to arbitrarily reparametrize time. Examples include true omnidirectional drive, differential drive, and a manipulator arm. Note that this requirement is looser than a prohibition on nonholonomic constraints.

Theorem 3: MBHP is resolution complete when controlling a SSC robot model. ■

Proof: The local planner is involved in two loops: an intra- and inter-cycle loop. The intra-cycle loop involves calls to the global planner to quickly provide a low-fidelity path to the goal. The inter-cycle loop operates more slowly and involves the robot moving and the local planner beginning anew from a slightly different start state. The path ultimately executed by the robot comprises a series of trajectory intervals generated by the slow inter-cycle loop. The quick intra-cycle path serves as a proxy (i.e. an existence proof) for the true path that is eventually executed.

Each planner is individually resolution complete over its own scope. Given a SSC motion model, the robot could at any time simply execute a path produced by the global planner. If no other path but the global plan exists, a resolution-complete local planner may continue to sample at finer action discretizations until an acceptably close approximation of the global path is reached[1]. Thus, for a SSC motion model, the completeness of the individual planners implies completeness of the resulting MBHP hierarchy. ■

It is worth noting that this proof holds only when the robots maintain a fixed communication graph. This stipulation occurs because of the difficulty in achieving consensus in the case of an arbitrarily changing graph topology. In the context of multirobot collision avoidance, the robots are only guaranteed sufficient time to find a new plan while executing their contingency plan, thus we feel this is a reasonable restriction.

We also restrict the proof to SSC robots. We do not mean to suggest that other motion models cannot be made complete within the MBHP framework. However, the challenge is that the global planner must generate paths that fulfill the role of a proxy for an eventual local path. In practice, it is difficult to make a global planner scalable while also generating approximable paths.

V. Collision Avoidance Properties

We now turn our attention towards the more general consideration of the distributed collision avoidance application described in Section III-B. In particular, we show that a variety of useful properties can be demonstrated for the whole system based on the completeness of its constituent motion planners.

Theorem 4: Any solution path returned by a (resolution- or probabilistically-) complete motion planner (or ensemble of planners) is free from deadlock, livelock, and starvation over the scope of its problem specification. ■

Proof: Coffman et al. [7] enumerate the four conditions necessary for deadlock to occur: mutual exclusion, resource holding, no preemption, and circular wait. In the case of a robot motion planner, the resource under contention is the set of points occupied by the volume of the robot in the task space over time. A complete planner is able to break a circular wait condition by preemption—for example, one robot moves out of the way of another.

[1]The local planner normally chooses not to execute the global path because it is capable of representing smoother, more optimal trajectories than the global grid path.

When a complete planner (or ensemble of planners) returns a plan, that plan gives a trajectory utilizing the full scope of the problem connecting the start and goal states. No situation that can be modeled by the planning problem will prevent the robot(s) from executing the trajectory in the joint state space until the goal state is reached. Thus, in the full state space, all robots have reached their desired location without livelock or starvation. ■

Corollary 1: A single SSC robot executing MBHP produces behavior that is free of deadlock and livelock. ■

Corollary 2: Multiple SSC robots executing MBHP with collision avoidance form an ensemble of planners whose behavior that is free of deadlock and livelock. ■

VI. Discussion

In this paper, we formalize the concept of an ensemble of motion planners that are decentralized but coupled to some degree. Such planners must coordinate in order to solve larger planning problems by combining each of their capabilities.

Decentralization often results in significant improvements in efficiency when compared to a single centralized planner that must solve the joint planning problem in the full state space. We show that these benefits can be gained without sacrificing completeness of the ensemble of planners. Completeness requires in essence only two properties: the combinatorics of completeness and communication suitable to reach consensus. The combinatorics of completeness require only that every combination of partial plans reported by each planner will eventually be tested. Completeness, in turn, provides a number of important guarantees for decentralized multi-robot algorithms. A complete planner avoids the problems of deadlock, livelock, and starvation.

In closing, it should be emphasized that not all planning problems are equally amenable to the decoupling approach enabled by an ensemble of planners. In the worst case, the countable combinatorics requirement can lead to memory usage that is exponential in the size of the planning problem. Such queries are better addressed in a centralized fashion.

The expansiveness property of an environment [12] provides an indication of the potential for decoupling the planning problem. In an expansive environment, many possible solutions exist, and so the problem of joining a set of partial plans diminishes in difficulty compared to the problem of generating the paths.

References

[1] J. Bang-Jensen and G. Gutin. *Digraphs: Theory, Algorithms and Applications*. Springer Monographs in Mathematics, London, 2009.

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

[3] K.E. Bekris, D.K. Grady, M. Moll, and L.E. Kavraki. Safe distributed motion coordination for second-order systems with different planning cycles. *International Journal of Robotics Research*, 31(2), February 2012 2012.

[4] R. A. Brooks and T. Lozano-Perez. A subdivision algorithm in configuration space for findpath with rotation. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-15(2): 224–233, March/April 1985.

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

[6] T. Chandra, R. Griesemer, and J. Redstone. Paxos made live – an engineering perspective. In *Proceedings of the 26th ACM Symposium on Principles of Distributed Computing*, 2007.

[7] E.G. Coffman, Jr., M.J. Elphick, and A. Shoshani. System deadlocks. *Computing Surveys*, 3(2), June 1971.

[8] M. Daily, J. Harris, D. Keirsey, D. Olin, D. Payton, K. Reiser, J. Rosenblatt, D. Tseng, and V. Wong. Autonomous cross-country navigation with the ALV. In *Proceedings of the International Conference on Robotics and Automation*, Philadelphia, PA, 1988.

[9] S. Dalibard, A. El Khouryz, F. Lamiraux, M. Taix, and J.P. Laumond. Small-space controllability of a walking humanoid robot. In *11th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 739–744, 2011.

[10] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[11] C. Dornhege, P. Eyerich, T. Keller, M. Brenner, and B. Nebel. Integrating task and motion planning using semantic attachments. In *Workshops at the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[12] D. Hsu, J.C. Latombe, and H. Kurniawati. On the probabilistic foundations of probabilistic roadmap planning. *International Journal of Robotics Research*, 25(7):627–643, 2006.

[13] L.P. Kaelbling and T. Lozano-Perez. Hierarchical planning in the now. In *Workshops at the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[14] R.A. Knepper and D. Rus. Pedestrian-inspired sampling-based multi-robot collision avoidance. In *Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication*, Paris, France, September 2012.

[15] R.A. Knepper, S.S. Srinivasa, and M.T. Mason. Hierarchical planning architectures for mobile manipulation tasks in indoor environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Anchorage, USA, May 2010.

[16] S. Koenig and M. Likhachev. D*lite. In *Proceedings of AAAI/IAAI*, pages 476–483, 2002.

[17] S. M. LaValle, M. S. Branicky, and S. R. Lindemann. On the relationship between classical grid search and probabilistic roadmaps. *International Journal of Robotics Research*, 23 (7/8):673–692, July/August 2004.

[18] S.M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[19] T. Lozano-Perez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, C-32(2):108–120, February 1983.

[20] T. Lozano-Perez, J.J. Jones, E. Mazer, P.A. O'Donnell, W.E.L. Grimson, P. Tournassoud, and A. Lanusse. Handey: A robot system that recognizes, plans, and manipulates. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 4, 1987.

[21] B. Marthi, S.J. Russell, and J. Wolfe. Angelic semantics for high-level actions. In *17th International Conference on Automated Planning and Scheduling (ICAPS)*, Providence, USA, 2007.

[22] C. McGann, E. Berger, J. Bohren, S. Chitta, B. P. Gerkey, S. Glaser, B. Marthi, W. Meeussen, T. Pratkanis, E. Marder-Eppstein, and M. Wise. Model-based, hierarchical control of a mobile manipulation platform. In *ICAPS Workshop on Planning and Plan Execution for Real-World Systems*, Thessaloniki, Greece, 2009.

[23] J. Snape, J. van den Berg, S. Guy, and D. Manocha. The hybrid reciprocal velocity obstacle. *IEEE Transactions on Robotics*, 27(4):696–706, 2011.