Centralized Path Planning for Multiple Robots: Optimal Decoupling into Sequential Plans

Jur van den Berg   Jack Snoeyink   Ming Lin   Dinesh Manocha
Department of Computer Science, University of North Carolina at Chapel Hill, USA.
E-mail: {berg, snoeyink, lin, dm}@cs.unc.edu

Abstract— In this paper we present an efficient algorithm for path planning for multiple robots. In our approach, the problem is decoupled into a set of sub-problems whose solutions can be sequentially executed, such that the dimension of the highest-dimensional sub-problem is *minimal* among all valid execution sequences. Our algorithm obtains this optimal sequence by planning only in configuration spaces of a dimension less than or equal to this minimum. This makes complicated multi-robot problems solvable, even if they involve many robots. Our approach relies on an external path planner for general configuration spaces, and inherits its properties; our approach is complete and applicable to robots of any kind with any number of degrees of freedom if the external planner is too. We implemented our algorithm, and prove its effectiveness by showing results on challenging scenarios involving 16 and 65 robots.[1]

I. Introduction

In this paper, we discuss the problem of path planning for multiple robots, which arises in different applications in Robotics. The objective is to move multiple robots in a common workspace from a given start configuration to a given goal configuration without mutual collisions and collisions with obstacles. We assume that an exact representation of the geometry of the robots and the workspace is given.

This problem has been studied extensively. Approaches are often characterized as *centralized* (a better term would be *coupled*) or *decoupled*: A coupled planner computes a path in a combined configuration space, which essentially treats the robots as a single combined robot. A decoupled planner may compute a path for each robot independently, then use a coordination diagram to plan collision-free trajectories for each robot along its path. Or it may plan a trajectory for each robot in order of priority and avoid the positions of previously planned robots, which are considered as moving obstacles.

Decoupled planners are generally faster, usually because fewer degrees of freedom are considered at one time. Unfortunately, they are usually not *complete* – some coupling may be necessary, as when two robots each have their goal as the other's start position, so a decoupled planner may not find a solution, even when one exists. Centralized or coupled planners, on the other hand, may be complete in theory, but they may need to work in configuration spaces of impractically high dimensions, regardless of how challenging the actual instance of the planning problem is – two robots in separate rooms would still be considered as a system with double the degrees of freedom, even though their tasks can be carried out independently without interference.

In this paper, we demonstrate an algorithm for multiple robot planning problems that decomposes any instance of multi-robot planning into a sequence of sub-problems with the minimum degree of coupled control. Informally, the control of two robots must be directly coupled if they must move at the same time to achieve their goals. The transitive closure of the direct coupling relationship is an equivalence relation that partitions the robots into classes that must be planned together as a composite. The degree of a composite robot is the sum of the number of degrees of freedom of the individual robots that are coupled.

We partition the robots into an ordered sequence of composite robots so that each composite can move from start to goal in turn, and *minimize* the maximum degree of all composite robots. If the problem instance has degree of coupling α, our algorithm is complete if we have access to an external general-purpose path planner that is complete for up to α degrees of freedom. Although the number of robots may appear exponentially in the combinatorial parts of the algorithm (though our experiments show that this worst case may be avoided), their degrees of freedom do not blow up the complexity of planning. Thus, our implementation is able to solve challenging scenarios which cannot be solved by traditional multi-robot planners. It is applicable to robots of any kind with any number of degrees of freedom, provided that our external planner is too.

After a brief review of related work in Section II, we define in Section III the notions of composite robots and execution sequences and constraints upon them, which we assume are returned from our external planner. In Section IV we present our algorithm and prove its properties, and in Section VI we discuss our experimental results. We conclude the paper in Section VII.

II. Related Work

Path planning for multiple robots has been extensively studied for decades. For the general background and theory of motion planning and coordination, we refer readers to [10, 13]. In this section, we review prior work that addresses similar problems as ours.

As mentioned earlier, prior work for multiple robots are often classified into *coupled* and *decoupled* planners. The coupled approaches aggregate all the individual robots into one

---

large composite system and apply single-robot motion planning algorithms. Much of classical motion planning techniques for exact motion planning, randomized motion planning and their variants would apply directly [10, 9, 11, 13].

Decoupled approaches plan for each robot individually and then perform a velocity tuning step in order to avoid collisions along these paths [29, 23, 18, 20, 22]. Alternatively, other schemes such as coordination graphs [15], or incremental planning [21] can help to ensure that no collisions occur along the paths. Prioritized approaches plan a trajectory for each robot in order of priority and avoid the positions of previously planned robots, which are considered as moving obstacles [7]. The choice of the priorities can have a large impact on the performance of the algorithm [27]. Some planners also search through a space of prioritizations [3, 4].

In addition, hybrid methods combine aspects of both coupled and decoupled approaches to create approaches that are more reliable or offer completeness but also scale better than coupled approaches [1, 2, 14, 26, 19, 5].

Geometric assembly problems initially seem closely related, especially when they speak of the number of "hands" or distinct motions needed to assemble and configuration of objects [16, 24, 17]. The various blocking graphs described in [28, 8] inspired the constraint graphs that we use. Differences can be seen on closer inspection: Assembly problems often restrict motions to simple translations or screws, and the aim is to create subassemblies that will then move together. (Our individual robots would need coordinated planning to move together as a subassembly.) Start positions for objects are usually uncomplicated, but assembly plans may need to be carried by some manipulator that must be able to reach or grasp the objects. In some sense, our algorithm combines these ideas: it captures constraints and then uses them to reduce the complexity of the motions that are planned.

III. DEFINITIONS AND PRELIMINARIES

Our multi-robot planning problem is formally defined as follows. The input consists of n robots, $r_1, \ldots, r_n$, and a common workspace in which these robots move (imagine a two- or three-dimensional scene with obstacles). The configuration space of robot r_i, denoted $\mathcal{C}(r_i)$, has dimension $\dim(r_i)$ equal to the number of degrees of freedom of robot r_i. Each robot r_i has a start configuration $s_i \in \mathcal{C}(r_i)$ and a goal configuration $g_i \in \mathcal{C}(r_i)$.

The task is to compute a path $\pi : [0,1] \in \mathcal{C}(r_1) \times \ldots \times \mathcal{C}(r_n)$, such that initially $\pi(0) = (s_1, \ldots, s_n)$, finally $\pi(1) = (g_1, \ldots, g_n)$, and at each intermediate time $t \in [0,1]$, with robots positioned at $\pi(t)$, no robot collides with an obstacle in the workspace or with another robot. At times we will refer to trajectories, which are the projections of a path into the space of a subset of robots.

The rest of this section gives precise definitions and properties for notions that we use to construct our algorithm for computing plans for composite robots with minimum degree of coupling.
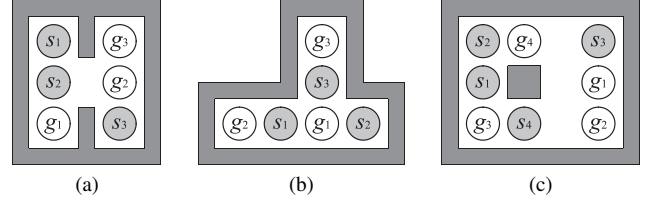

Fig. 1. Three simple example problems for circular planar robots, with start (gray) and goal (white) configurations shown. The dark gray delineates the static obstacles. Solution sequences for (a) = $[r_3, r_2, r_1]$ and (b) = $[r_3, r_1 r_2]$. Instance (c) has four solution sequences, $[r_2, r_1, r_3, r_4], [r_2, r_1, r_4, r_3], [r_2, r_4, r_1, r_3]$, and $[r_4, r_1, r_2, r_3]$.

A. The Coupled Relation from a Solution Path

Consider a path π that solves a multi-robot planning problem, as defined above. For each robot r_i, we define an *active interval* $\tau_i \subset [0,1]$ as the open interval from the first time r_i leaves its start position to the last time r_i reaches its goal position.

Definition 1 (Coupled relation). Two robots r_i, r_j are *directly coupled* if their active intervals intersect, $\tau_i \cap \tau_j \neq \emptyset$. The transitive closure of this relation is an equivalence relation, and we say that robots in the same equivalence class are *coupled*.

On the interval $[0,1]$, the equivalence classes of the coupled relation determine the connected subsets of the union of all active intervals. If we list the equivalence classes in order of increasing time, we get a sequence of coupled or composite robots that can be abstracted from a specific path as an *execution sequence*, as we define in the next subsection.

The *degree of coupling of a solution path* is the maximum of the sum of the degrees of freedom in any connected set of active intervals; these are the maximum number of degrees of freedom that need be considered simultaneously by any coupled planner to construct or validate this sequential plan. The *degree of coupling of a planning instance* is the minimum degree of coupling of any solution path. Our algorithm finds a solution path that achieves the minimum degree of coupling α for the given instance, by using an external coupled planner on problems with at most α degrees of freedom. While hard, puzzle-like problem instances may still require a full-dimensional configuration space, we argue that many "realistic" problems, even those involving many robots, can be solved with much lower dimensional planning.

B. Execution Sequences

In our algorithm, we will decompose the multi-robot motion planning problem into lower-dimensional sub-problems that can be *sequentially* executed. In these sub-problems, some individual robots will be coupled and considered as composite robots by the lower-dimensional planner.

Definition 2 (Composite Robot). We define a *composite* robot $R \subseteq \{r_1, \ldots, r_n\}$ as a subset of the n robots that is treated as one coupled or composite body. R's configuration space, $\mathcal{C}(R)$, is the Cartesian product of the configuration spaces of the robots in R, its dimension is the sum of the degrees of

freedom of the robots in R: $\dim(R) = \sum_{r_i \in R} \dim(r_i)$, and its active interval is the smallest interval that contains the union of the active intervals of all $r_i \in R$.

Since individual robots can also be thought of as composite, we will tend to omit the word "composite" and just say "robot." When we want to emphasize the composite nature of a robot, we concatenate the individual robots it consists of. For example $\{r_1, r_2\} = r_1 r_2$.

Definition 3 (Execution Sequence). We define an *execution sequence* S as an ordered partition of the n robots into a sequence $S = (R_1, \ldots, R_k)$ of composite robots, such that $R_1 \cup \cdots \cup R_k = \{r_1, \ldots, r_n\}$ and $R_i \cap R_j = \emptyset$ for $i \neq j$.

An execution sequence is *valid* if it is the sequence of equivalence classes of the *coupled* relation for some solution path (with no collisions).

We call a valid execution sequence a *solution sequence*.

Our algorithm will find a solution sequence with the minimum number of degrees of freedom, so we define the dimension $\dim(S)$ of a solution sequence $S = (R_1, \ldots, R_k)$ as the dimension of the largest composite robot in the solution sequence: $\dim(S) = \max_{R_i \in S} \dim(R_i)$. An *optimal solution sequence* is a solution sequence S^* with minimal dimension among all solution sequences: $S^* = \arg\min_S \dim(S)$.

Fig. 1 illustrates three example problems for 2-D circular robots in the plane, each robot r_i must find a path from the gray start postion s_i to the corresponding white goal position g_i without collisions. A valid solution path for (a) is to first move r_3, then r_2 and then r_1 – i.e., execution sequence $S = (r_3, r_2, r_1)$ solves the problem. Hence, the 6-D configuration space of problem (a) can be decomposed into three sequential 2-D sub-problems. For problem (b) there is no solution by moving individual robots. But a valid solution path could first move r_3, and then move r_1 and r_2 simultaneously as a composite robot in a coordinated effort to reach their goal – i.e., execution sequence $S = (r_3, r_1 r_2)$ solves the problem. Hence, problem (b) can be decomposed into one 2-D sub-problem and one 4-D sub-problem. Problem (c) has four possible execution sequences, all of which have either r_4 or both r_1 and r_2 moving before r_3.

C. Order Constraints from a Robot R

Generalizing from these examples, we can observe that valid execution sequences depend only on the start or goal positions of inactive robots.

Observation 4. Execution sequence S is valid if, for all $i \in [1, k]$, robot $R_i \in S$ can move from its start to its goal without collisions, even when the *goal* configurations of robots in $\{R_1, \ldots, R_{i-1}\}$ and the *start* configurations of robots in $\{R_{i+1}, \ldots, R_k\}$ have been added to the obstacles.

By a thought experiment, let's develop notation for constraints on the ordering of robots in solution sequences for specific trajectories before giving the formal definition. Suppose a specific trajectory for (individual or composite) robot

R has a collision with the goal configuration of robot r_j. We then write $R \prec r_j$ to indicate that R must either complete its motion before r_j begins, or R and r_j will be coupled since their active intervals overlap. Similarly, if the trajectory collides with the start configuration of robot r_k, we may write $r_k \prec R$. We collect all collisions with a single trajectory into a conjunction, then write a disjunction of the conjunctions for all possible trajectories, and simplify the resulting expression in disjunctive normal form (DNF), which we denote $\mathcal{P}(R)$. For example, if some trajectory for R collides with no start or goal positions, then $\mathcal{P}(R) = \top$ ('true'), and In general, we need to keep only the *minterms* – those conjunctions that do not contain another as a subset.

For example, in Fig. 1(c) robot r_4 can reach its goal without going through any query configurations of other robots, so $\mathcal{P}(r_4) = \top$ ('true'). If, due to static obstacles, some robot has no path to the goal at all, we could write its expression as $\bot$ ('false'). Robot r_1 can either move through the goal configuration of r_3 and the start configuration of r_4 to reach its goal, *or* it can move through the start configuration of r_2 and the goal configuration of r_4. Hence:

$$\mathcal{P}(r_1) = (r_2 \prec r_1 \wedge r_1 \prec r_4) \vee (r_1 \prec r_3 \wedge r_4 \prec r_1).$$

In Fig. 1(b), robot r_2 has to move through both the start and the goal configuration of robot r_1. This gives the constraint $\mathcal{P}(r_2) = r_2 \prec r_1 \wedge r_1 \prec r_2$, which we may abbreviate as $r_1 \sim r_2$. This means that r_1 and r_2 need to be active simultaneously, so their motion must be planned as a composite robot $r_1 r_2$, for which $\mathcal{P}(r_1 r_2) = (r_3 \prec r_1 r_2)$.

The tilde abbreviation, and the fact that our ordering relation is transitive, allows us to rewrite each conjunction of $\mathcal{P}(R)$ in the following *atomic form*: we arbitrarily choose a representative $r \in R$, replace each capital R with the representative r, and AND the expression $\bigwedge_{r' \in R} r \sim r'$. Thus, for Fig. 1(b), we have the atomic form $\mathcal{P}(r_1 r_2) = r_3 \prec r_1 \wedge r_1 \sim r_2$.

Each conjunction has a natural interpretation as a directed graph in which robots are vertices and directed edges indicate $\prec$ relations. By transitivity, any directed path from r_i to r_j indicates a relation $r_i \prec r_j$ and any pair of vertices in the same strongly connected component are related by $\sim$. We use such constraint graphs in our implementation in Section IV-A.

Two properties of these constraint expressions are easy to observe. First, by construction,

Property 5. For each atomic constraint $r \prec r'$ in the constraint expression $\mathcal{P}(R)$ of (composite) robot R, either $r \in R$ or $r' \in R$.

Second, because any trajectory for a larger composite robot includes trajectories for subsets, we can observe that the constraints on the larger robot imply those on the smaller.

Property 6. If $R' \supseteq R$, then $\mathcal{P}(R') \Rightarrow \mathcal{P}(R)$.

This means, for instance, that if a robot r_i needs to move before a robot r_j, then any composite robot R involving r_i needs to move before r_j. This is an important property, as it allows us to obtain the constraints on the execution sequence

iteratively, starting with the robots of smallest dimension (fewest degrees of freedom).

D. Constraints from an Execution Sequence

If we AND the constraints for each robot in an execution sequence, we get the expression that must be satisfied for it to be a solution sequence – a sequence in which each composite robot has a valid trajectory.

Lemma 7. An execution sequence $S = (R_1, \ldots, R_k)$ is a *solution sequence* if and only if S satisfies the constraint expression $\mathcal{P}(R_1) \wedge \cdots \wedge \mathcal{P}(R_k)$.

Proof: If S satisfies $\mathcal{P}(R_i)$, (composite) robot R_i can reach its goal configuration without moving through any of the *goal* configurations of robots in $\{R_1, \ldots, R_{i-1}\}$, and without moving through any of the *start* configurations of robots in $\{R_{i+1}, \ldots, R_k\}$. At the moment of execution of R_i, all robots in $\{R_1, \ldots, R_{i-1}\}$ reside at their goal configuration, as they have already been executed, and all robots in $\{R_{i+1}, \ldots, R_k\}$ reside at their start configuration, as they have not yet been executed. Hence, R_i can validly move to its goal configuration. If S does not satisfy $\mathcal{P}(R_i)$, (composite) robot R_i either has to move through any of the *goal* configurations of robots in $\{R_1, \ldots, R_{i-1}\}$, or through any of the *start* configurations of robots in $\{R_{i+1}, \ldots, R_k\}$ in order to reach its goal. As all robots in $\{R_1, \ldots, R_{i-1}\}$ reside at their goal configuration and all robots in $\{R_{i+1}, \ldots, R_k\}$ reside at their start configuration at the moment of execution of R_i, it is not possible for R_i to validly reach its goal. □

Notice that because we AND the DNF expressions for the robots of an execution sequence, we can convert the resulting expression back to DNF by distributing across the new ANDs. The conjunctions of the resulting disjunction simply take one conjunction from the DNF for each robot in the execution sequence, so any sequence that satisfies the expression can directly seen to satisfy the DNF expressions for each constituent robot.

E. The CR planner for low-dimensional sub-problems

Now, let us postulate a *CR planner* (coupled or composite robot planner), that given the workspace, with start and goal configurations for robots $\{r_1, r_2, \ldots, r_n\}$, and a small subset R of these robots, returns the DNF expression $\mathcal{P}(R)$ with each clause in atomic form.

As a proof of existence, we can construct a CR planner using as a black box any complete planner that can determine feasibility for the composite robot R in a fixed workspace. Simply apply the black box to at most $4^{n-|R|}$ instances, with each other robot $r' \notin R$ added as an obstacle in its start position ($r \prec r'$), its goal position ($r' \prec r$), both ($\top$), or neither ($r \sim r'$). Each feasible path found adds a conjunction of all its constraints to the DNF expression, which is simplified and put in atomic form. Our actual planner, described in Section V, will be less costly.

In the next section, we present an algorithm that efficiently finds an optimal solution sequence S^* to solve the multi-robot

Iteration 1. $L = \emptyset$, $R_{\min} = r_1$, $\mathcal{P}(r_1) = (r_2 \prec r_1 \wedge r_1 \prec r_4) \vee (r_1 \prec r_3 \wedge r_4 \prec r_1)$.

Iteration 2. $L = \{r_1\}$, $R_{\min} = r_2$, $\mathcal{P}(r_2) = r_2 \prec r_4 \vee (r_1 \prec r_2 \wedge r_2 \prec r_3 \wedge r_4 \prec r_2)$.

Iteration 3. $L = \{r_1, r_2\}$, $R_{\min} = r_3$, $\mathcal{P}(r_3) = r_4 \prec r_3 \vee (r_3 \prec r_4 \wedge r_2 \prec r_3 \wedge r_1 \prec r_3)$.

Iteration 4. $L = \{r_1, r_2, r_3\}$, $R_{\min} = r_4$, $\mathcal{P}(r_4) = \top$, so no change to graphs.

The algorithm terminates. $L = \{r_1, r_2, r_3, r_4\}$ and the first conjunction in E has all of its strongly connected components in L. The solution sequence returned is $S = (r_2, r_1, r_4, r_3)$.
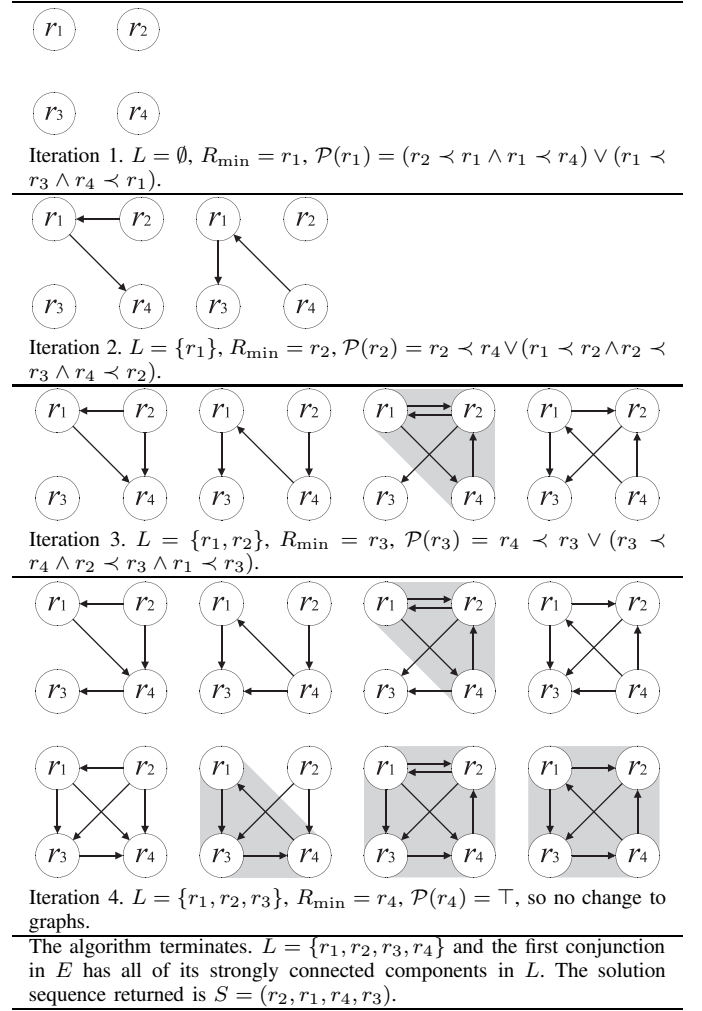
Fig. 2. An illustration of the steps of our algorithm on the problem of Fig. 1(c). In each iteration the constraints $\mathcal{P}(R)$ of a new robot are incorporated into E. We show the conjunctions in E as a set of constraint graphs. Strongly connected components in the constraint graphs are indicated by a gray background. Initially $E = \{\top\}$, which corresponds to one empty constraint graph.

planning problem. Our algorithm is able to do this by planning only in configuration spaces whose dimension is less than or equal to $\dim(S^*)$.

IV. INCREMENTAL DISCOVERY OF COUPLING

In this section we explain how to use the CR planner, which produces constraints on execution sequences induced by small subsets of robots, to find the lowest degree of coupling that will solve the given instance of a multi-robot planning problem. Our algorithm incrementally calls the planner on higher and higher degree sub-problems, using the discovered constraints to determine what robots must be coupled. First, we describe the constraint graph, a data structure for the constraint expressions we have collected.

A. Constraint Graphs

In our algorithm, we maintain the constraints we have obtained so far in a constraint expression E. We represent E in

disjunctive normal form, i.e. as a disjunction $E = J_1 \vee J_2 \vee \cdots$ of conjunctions J_i. Each conjunction J can be represented by a *graph* $G(J)$, which we call a *constraint graph*. A constraint graph has n nodes, one for each robot r_i, and a set of *directed* edges that indicate constraints on the order of execution of the robots. That is, for each atomic constraint $r_i \prec r_j$ in J, there is an edge from the node of r_i to the node of r_j in $G(J)$ (see, for example, Fig. 2). If $J = \top$, the corresponding constraint graph $G(\top)$ does not contain any edges.

If a constraint graph contains a cycle, there is a contradiction among the constraints. This means that the involved robots need to be coordinated as a composite robot in order to find a solution. To be more precise, the set of nodes (robots) in a graph is partitioned into a set of *strongly connected components*. A strongly connected component is a maximal set of nodes that are *strongly connected* to each other; two nodes r_i and r_j are strongly connected if there is a path in the graph both from r_i to r_j and from r_j to r_i. By definition, each node is strongly connected to itself.

Let $G^{SCC}(J)$ denote the *component graph* of $G(J)$, which contains a node for each strongly connected component of $G(J)$ and a directed edge from node R to node R' if there is an edge in $G(J)$ from any $r \in R$ to any $r' \in R'$. Note that G^{SCC} is a directed *acyclic* graph. Each node in $G^{SCC}(J)$ corresponds to a (composite) robot consisting of the robots involved in the strongly connected component. Topologically sorting $G^{SCC}(J)$ gives an execution sequence $S(J)$ of composite robots. Trivially, the following holds:

Corollary 8. If $G(J)$ is a constraint graph corresponding to conjunction J, then $S(J)$ is an execution sequence that satisfies J.

B. Incrementally Building the Execution Sequence

To build the expression sequence for an instance of multi-robot planning, our algorithm primarily maintains a DNF constraint expression E in the form of constraint graphs for its conjunctions J_i (if there are no conjunctons, $E = \bot$). Our algorithm also maintains a list L of the (composite) robots that have been passed to the CR planner and whose constraints $\mathcal{P}(R)$ have been incorporated into E.

Initially, $E = \{\top\}$, as we begin with no constraints. Now, iteratively, we select the (composite) robot $R_{\min}$ that has the smallest dimension among all (composite) robots for which we have not yet planned in the execution sequences $S(J)$ of all conjunctions $J \in E$:

$$R_{\min} = \underset{R \in \bigcup_{J \in E} S(J) \setminus L}{\arg\min} \dim(R).$$

Next, the CR planner is invoked on $R_{\min}$; it returns the set of constraints $\mathcal{P}(R_{\min})$. For *each* conjunction J in E for which $R_{\min} \in S(J)$, we do the following:

- Let $F = J \wedge \mathcal{P}(R_{\min})$, and transform F into disjunctive normal form. (Note that for each conjunction J' in F the following holds: $J' \Rightarrow J$ and $J' \Rightarrow \mathcal{P}(R_{\min})$.)
- Remove J from E and add the conjunctions of F to E (we replace J in E by the conjunctions of F).

The constraints of $\mathcal{P}(R_{\min})$ have now been incorporated into E, so we add $R_{\min}$ to the set L.

This procedure repeats until either $E = \emptyset$, in which case there is no solution to the multi-robot planning problem, *or* there exists a conjunction $J_{\text{sol}} \in E$ for which all composite robots $R \in S(J_{\text{sol}})$ have been planned for and are in L. In this case $S(J_{\text{sol}})$ is an *optimal solution sequence*, which we will prove below. In Fig. 2, we show the working of our algorithm on the example of Fig. 1(c).

C. Analysis

Here we prove that the above algorithm gives an optimal solution sequence:

Lemma 9. In each iteration of the algorithm, the constraints $\mathcal{P}(R_{\min})$ of composite robot $R_{\min}$ are incorporated into E. Right after the iteration, the following holds for all conjunctions $J \in E$: if $R_{\min} \in S(J)$ then $J \Rightarrow \mathcal{P}(R_{\min})$.

Proof: When we incorporate $\mathcal{P}(R_{\min})$ into E, all $J \in E$ for which $R_{\min} \in S(J)$ are replaced in E by $F = J \wedge \mathcal{P}(R_{\min})$. Now, all conjunctions $J' \in E$ for which $R_{\min} \in S(J')$ must be in F. Hence $J' \Rightarrow F$ and as a result $J' \Rightarrow \mathcal{P}(R_{\min})$. $\square$

Lemma 10. In each iteration of the algorithm, the constraints $\mathcal{P}(R_{\min})$ of composite robot $R_{\min}$ are incorporated into E. Its dimension is *greater than or equal to* the dimensions of all composite robots in L whose constraints were incorporated before: $\dim(R_{\min}) \geq \max_{R \in L} \dim(R)$.

Proof: Assume the converse is true: let R be the composite robot whose constraints were incorporated in the previous iteration, and let $\dim(R_{\min}) < \dim(R)$. Let $J_{\min}$ be the conjunction in E for which $R_{\min} \in S(J_{\min})$, Then, in the iteration R was selected, $J_{\min}$ did not exist yet in E, otherwise $R_{\min}$ would have been selected (our algorithm always selects the lowest-dimensional composite robot). This means that right before $\mathcal{P}(R)$ was incorporated, there was a $J \in E$ for which $R \in S(J)$ and $R_{\min} \notin S(J)$ that was replaced by $F = J \wedge \mathcal{P}(R)$ such that $J_{\min} \in F$. This means that one or more edges were added to $G(J)$ which caused the robots in $R_{\min}$ to form a strongly connected component in $G(J_{\min})$. Hence, these edges must have been between nodes corresponding to robots in $R_{\min}$. As these edges can only have come from $\mathcal{P}(R)$, this means, by Property 5, that $R \cap R_{\min} \neq \emptyset$. However, R forms a strongly connected component in $G(J)$ as $R \in S(J)$, so then $R \cup R_{\min}$ must also be a strongly connected component in $G(J_{\min})$. $R \cup R_{\min}$ and $R_{\min}$ can only be strongly connected components at the same time if $R \subset R_{\min}$. However, this means that $\dim(R) < \dim(R_{\min})$, so we have reached a contradiction. $\square$

Lemma 11. After each iteration of the algorithm, the following holds for all composite robots $R \in L$ whose constraints have been incorporated into E: for all conjunctions $J \in E$, if $R \in S(J)$ then $J \Rightarrow \mathcal{P}(R)$.

Proof: Lemma 9 proves that right after $\mathcal{P}(R)$ is incorporated, that for all conjunctions $J \in E$, if $R \in S(J)$ then $J \Rightarrow$

$\mathcal{P}(R)$. Now, we show that after a next iteration in which the constraints $\mathcal{P}(R')$ of another (composite) robot R' are incorporated, this still holds for R. Let J be a conjunction in E after $\mathcal{P}(R')$ is incorporated for which $R \in S(J)$. Now, either J already existed after the previous iteration, in which case $J \Rightarrow \mathcal{P}(R)$, or there existed a $J' \in E$ after the previous iteration that was replaced by $F = J' \wedge \mathcal{P}(R')$ such that $J \in F$. In the latter case, either $J' \Rightarrow \mathcal{P}(R)$, in which case also $J \Rightarrow \mathcal{P}(R)$, or $J' \not\Rightarrow \mathcal{P}(R)$ and $R \notin S(J')$. In the latter case, one or more edges must have been added to $G(J')$ by $\mathcal{P}(R')$ that caused the robots in R to form a strongly connected component in $G(J)$. Along similar lines as in the proof of Lemma 10, this means that $\dim(R') < \dim(R)$. However, by Lemma 10 this is not possible, as $\mathcal{P}(R')$ was incorporated later than $\mathcal{P}(R)$. The above argument can inductively be applied to all (composite) robots $R \in L$. $\qquad\square$

Theorem 12 (Correctness). The execution sequence S returned by the above algorithm is a solution sequence.

Proof: Let J_{sol} be the conjunction whose execution sequence $S(J_{\text{sol}}) = (R_1, \ldots, R_k)$ is returned by the algorithm. Then, all composite robots $R_i \in S(J_{\min})$ are also in L. By Lemma 11, $J_{\text{sol}} \Rightarrow \mathcal{P}(R_1) \wedge \cdots \wedge \mathcal{P}(R_k)$. Hence, by Corollary 8 and Lemma 7, $S(J_{\min})$ is a solution sequence. $\qquad\square$

Theorem 13 (Optimality). The execution sequence S returned by the above algorithm is an *optimal* solution sequence.

Proof: Sketch: Consider any other solution path π', and its execution sequence S' that comes from the equivalence classes of the coupled relation defined in Section III. This execution sequence satisfies the contraints induced by its robots, so the only way it could not have been discovered is for a different sequence of composite robots to be formed. Since the degree grows monotonically, and at all times the robot of lowest dimesion/degree is added to the plan, the execution sequence that is found by the algorithm cannot have larger degree. $\square$

Corollary 14 (Efficiency). The dimension of the highest-dimensional configuration space our algorithm plans in is equal to the dimension of the highest-dimensional (composite) robot in an *optimal* solution sequence.

V. IMPLEMENTATION AND OPTIMIZATION

In this section we describe some details of the implementation of our algorithm. We first describe how we maintain the set of constraints E in our main algorithm. We then describe how we implemented the CR planner that gives the constraints $\mathcal{P}(R)$ for a given composite robot R.

A. Main Algorithm

The implementation of our algorithm largely follows the algorithm as we have described in Section IV-B. We maintain the logical expression E in disjunctive normal form $J_1 \vee J_2 \vee \cdots$ as a set of constraint graphs $\{G(J_1), G(J_2), \ldots\}$. Each graph $G(J)$ is stored as an $n \times n$ *boolean matrix*, where a 1 at position (i, j) corresponds to an edge between r_i and r_j in $G(J)$. A boolean matrix can efficiently be stored

in memory, as each of its entries only require one bit. An operation $J \wedge J'$ of conjunctions J and J' can efficiently be performed by computing the bitwise-*or* of its corresponding boolean matrices. Also, checking whether $J \Rightarrow J'$ is easy; it is the case when the bitwise-or of the boolean matrices of $G(J)$ and $G(J')$ is equal to the boolean matrix of $G(J)$.

All graphs $G(J)$ are stored in *transitively closed* form. The transitive closure of a boolean matrix can efficiently be computed using the Floyd-Warshall algorithm [6]. Given a boolean matrix of a transitively closed graph $G(J)$, it is easy to infer the strongly connected components of $G(J)$.

B. Composite Robot Planner

The constraints $\mathcal{P}(R)$ for a (composite) robot R are obtained by path planning between the start configuration of R and the goal configuration of R in its configuration space $\mathcal{C}(R)$, and by considering through which query configurations of other robots R need to move. For our implementation, we sacrifice some completeness for practicality, by discretizing the configuration space $\mathcal{C}(R)$ into a *roadmap* $RM(R)$. Our discretization is similar to the one used for the planner presented in [26]. We sketch our implementation here.

Prior to running our algorithm, we construct for each *individual* robot r_i a roadmap $RM(r_i)$ that covers its configuration space $\mathcal{C}(r_i)$ well. Let us assume that the start configuration s_i and the goal configuration g_i of robot r_i are present as vertices in $RM(r_i)$. Further, each vertex and each edge in $RM(r_i)$ should be collision-free with respect to the static obstacles in the workspace. The roadmaps $RM(r_i)$ can be constructed by, for instance, a Probabilistic Roadmap Planner [9]. They are reused any time a call to the CR planner is made.

The (composite) roadmap $RM(R)$ of a composite robot $R = \{r_1, \ldots, r_k\}$ is defined as follows. There is a *vertex* $(x_1, \ldots, x_k)$ in $RM(R)$ if for all $i \in [1, k]$ x_i is a node in $RM(r_i)$, and for all $i, j \in [1, k]$ (with $i \neq j$) robots r_i and r_j configured at their vertices x_i and x_j, respectively, do not collide. There is an *edge* in $RM(R)$ between vertices $(x_1, \ldots, x_k)$ and $(y_1, \ldots, y_k)$ if for exactly one $i \in [1, k]$ $x_i \neq y_i$ and x_i is connected to y_i by an edge in $RM(r_i)$ that is not blocked by any robot $r_j \in R$ (with $i \neq j$) configured at x_j. This composite roadmap is not explicitly constructed, but explored implicitly while planning.

Now, to infer the constraints $\mathcal{P}(R)$ for a (composite) robot R, we plan in its (composite) roadmap $RM(R)$ using an algorithm very similar to Dijkstra's algorithm. However, instead of computing distances from the start configuration for each vertex, we compute a constraint expression $\mathcal{P}(x)$ for each vertex x. The logical implication "$\Rightarrow$" takes the role of "$\geq$". The algorithm is given in Algorithm 1.

Note that unlike Dijkstra's algorithm, this algorithm can visit vertices multiple times. In the worst case, this algorithm has a running time exponential in the total number of robots n, as the definition of $\mathcal{P}(R)$ in Section III-E suggests. However, in line 16 of Algorithm 1 many paths are pruned from further exploration, which makes the algorithm tractable for most practical cases. The planner is sped up by ordering the vertices

Algorithm 1 $\mathcal{P}(R)$

1: Let $s, g \in RM(R)$ be the start and goal configuration of R.
2: **for all** vertices x in $RM(R)$ **do**
3:      $\mathcal{P}(x) \leftarrow \perp$
4: $\mathcal{P}(s) \leftarrow \bigwedge_{r_i, r_j \in R} r_i \sim r_j$
5: $Q \leftarrow \{s\}$
6: **while not** (priority) queue Q is empty **do**
7:      Pop front vertex x from Q.
8:      **if not** $\mathcal{P}(x) \Rightarrow \mathcal{P}(g)$ **then**
9:          **for all** edges (x, x') in $RM(R)$ **do**
10:             $C \leftarrow \mathcal{P}(x)$
11:             **for all** robots $r_i \notin R$ **do**
12:                 **if** robot r_i configured at s_i "blocks" edge (x, x') **then**
13:                     $C \leftarrow C \wedge r_i \prec R$
14:                 **if** robot r_i configured at g_i "blocks" edge (x, x') **then**
15:                     $C \leftarrow C \wedge R \prec r_i$
16:             **if not** $C \Rightarrow \mathcal{P}(x')$ **then**
17:                 $\mathcal{P}(x') \leftarrow \mathcal{P}(x') \vee C$
18:                 **if** $x' \notin Q$ **then**
19:                     $Q \leftarrow Q \cup \{x'\}$
20: **return** $\mathcal{P}(g)$

Fig. 3. (a) An environment with sixteen robots that are shown in their start configuration. Each robot r_i has to exchange positions with robot $r_{((i+8) \bmod 16)}$, i.e. the robot on the opposite side of the room. (b) An environment with a variable number of robots n and a variable degree of coupling α. The robots in each column need to reverse their order as shown for the left column. (c) A randomly generated environment with 65 robots. The start and goal configurations are shown in light gray and white, respectively.

in the queue Q according to the partial ordering defined by the implication relation "$\Leftarrow$" on their constraint expressions, such that vertices whose constraint expressions are implied by other constraint expressions are explored first.

VI. RESULTS

We report results of our implementation on the three scenarios shown in Fig. 3. These are challenging scenarios, each for their own reason. Scenario (a) has relatively few robots, but a potentially high degree of coupling among them. Scenario (c) has many robots, but with a low degree of coupling. Problems of the first type are traditionally the domain of coupled planners; decoupled planners are likely to fail in these cases. Problems of the second type are the domain of decoupled methods; for a coupled planner there would be too many robots to be able to compute a solution. In this section, we show that our method copes well with both. In scenario (b), we vary both the number of robots n and the degree of coupling α, and study quantitatively how our algorithm performs in these varying circumstances.

Scenario (a) has sixteen robots $r_0, \ldots, r_{15}$. Each robot r_i has to exchange positions with robot $r_{((i+8) \bmod 16)}$, i.e. the robot on the opposite side of the room. There is very limited space to maneuver. Traditional multi-robot planners are unlikely to solve this problem. A decoupled two-stage planner will only succeed if the paths in the first stage are chosen exactly correct, which is not likely. A prioritized approach will only work if the correct prioritization is chosen. Also coupled planners are probably not able to solve this problem, because planning in a 32-dimensional composite configuration space is generally not feasible. Our method, on the other hand, solved this problem in 2.57 seconds on an Intel Core2Duo P7350 2GHz with 4GByte of memory. It returned the optimal solution sequence $(r_0 r_7 r_8 r_{15}, r_1 r_9, r_2 r_{10}, r_3 r_{11}, r_4 r_5 r_{12} r_{13}, r_6 r_{14}, r_7 r_{15})$. Our algorithm planned eight times for one robot, four times for

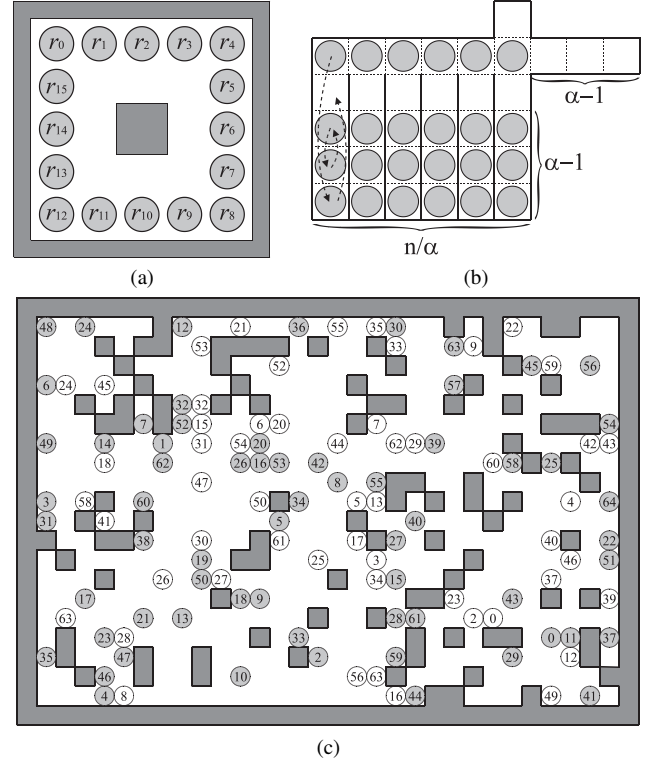two robots and twice for four robots in order to achieve this result.

Scenario (c) involves a randomly generated environment containing as many as 65 robots, giving a full composite configuration space of 130 dimensions. Each robot was assigned a random start and goal configuration. Our method returned a solution sequence solely containing individual robots. That means that our algorithm found a solution by only planning in the 65 two-dimensional configuration spaces of each of the robots. Even though the number of robots is high, the degree of coupling in this example is very low. Our algorithm exploits this; it solved this example in only 73 seconds. After the last iteration of the algorithm, the constraint expression E contained 4104 conjunctions, and the conjunction that provided the solution sequence contained 27 atomic constraints.

Experiments on scenario (b) show how our algorithm performs for a varying number of robots and degree of coupling. The scenario is designed such that it cannot be solved by decoupled planners. We report results in Fig. 4. In the first column, we set the degree of coupling equal to the number of robots, i.e. the problem is fully coupled. In this case, the planner fails for seven or more robots, despite the small size of the workspace. When the degree of coupling is lower, our approach is able to solve problems for a much higher number of robots in reasonable running times. Analysis of the results

$\alpha = n$		$\alpha = 2$		$\alpha = 3$		$\alpha = 4$	
n	time	n	time	n	time	n	time
5	1.39	20	0.30	27	18.8	20	41.7
6	44.0	22	0.69	30	38.8	24	167
7	n/a	24	2.15	33	75.6	28	542
		26	7.69	36	146	32	1254
		28	42.4	39	287	36	3356
		30	261	42	672	40	7244

Fig. 4. Performance results on scenario (b). Running times are in seconds.

indicate that (for constant degree of coupling) the running time increases polynomially with the number of robots for experiments with up to approximately 30 robots. There is a very small exponential component (due to combinatorics), which starts to dominate the running time for 30 or more robots. Nontheless, these results show that our approach is able to solve problems which could not be solved by either fully coupled planners or decoupled planners.

VII. CONCLUSION

In this paper, we have presented a novel algorithm for path planning for multiple robots. We have introduced a measure of "coupledness" of multi-robot planning problem instances, and we have proven that our algorithm computes the optimal solution sequence that has the minimal degree of coupling. Using our implementation, we were able to solve complicated problems, ranging from problems with a relatively few robots and high degree of coupling to problems with many robots and a low degree of coupling.

The *quality* of our solutions, usually defined in terms of arrival times of the robots [12], is not optimal as the computed robot plans are executed sequentially. An idea to improve this is to use a traditional *prioritized* planner in a post-processing step, and have it plan trajectories for the (composite) robots in order of the solution sequence our algorithm computes.

Another idea for future work is to exploit *parallelism* to increase the performance of our algorithm. Our current algorithm is iterative, but it seems possible that planning for different (composite) robots can be carried out in parallel.

A limitation of our current approach is that it computes a sequence of complete robot plans only (i.e. plans that go from start to goal). This causes more coupling than what would seem necessary for some problem instances. For example if three robots block the centers of a long hall that a fourth robot is trying to traverse, then all four would be coupled even though the three could each move out of the way in turn. Even though the three have no direct coupling, the are coupled by transitive closure through the fourth. If we add *intermediate goals* for the fourth, however, and use our ordering constraints to specify that these intermediate configurations must be visited in order, then it is easy to reduce the degree of coupling. Thus, the selection and placement of intermediate goals seems a fruitful topic for further study.

REFERENCES

[1] R. Alami, F. Robert, F. Ingrand, S. Suzuki. Multi-robot cooperation through incremental plan-merging. *Proc. IEEE Int. Conference on Robotics and Automation*, pp. 2573–2579, 1995.

[2] B. Aronov, M. de Berg, F. van der Stappen, P. Švestka, J. Vleugels. Motion planning for multiple robots. *Discrete and Computational Geometry* 22(4), pp. 505–525, 1999.

[3] M. Bennewitz, W. Burgard, S. Thrun. Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots. *Robotics and Autonomous Systems* 41(2), pp. 89–99, 2002.

[4] S. Buckley. Fast motion planning for multiple moving robots. *Proc. IEEE Int. Conference on Robotics and Automation*, pp. 322–326, 1989.

[5] C. Clark, S. Rock, J.-C. Latombe. Motion planning for multiple robot systems using dynamic networks. *Proc. IEEE Int. Conference on Robotics and Automation*, pp. 4222–4227, 2003.

[6] T. Cormen, C. Leiserson, R. Rivest. *Introduction to Algorithms*. The MIT Press, 1998.

[7] M. Erdmann, T. Lozano-Perez. On multiple moving objects. *Proc. IEEE Int. Conference on Robotics and Automation*, pp. 1419–1424, 1986.

[8] D. Halperin, J.-C. Latombe, R. Wilson. A general framework for assembly planning: The motion space approach. *Algorithmica* 26(3-4), pp. 577–601, 2000.

[9] L. Kavraki, P. Švestka, J.-C. Latombe, M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robotics and Automation* 12(4), pp. 566–580, 1996.

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

[11] S. LaValle, J. Kuffner. Rapidly-exploring random trees: Progress and prospects. *Workshop on the Algorithmic Foundations of Robotics*, 2000.

[12] S. LaValle, S. Hutchinson. Optimal motion planning for multiple robots having independent goals. *IEEE Trans. on Robotics and Automation* 14(6), pp. 912–925, 1998.

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

[14] T-Y. Li, H-C. Chou. Motion planning for a crowd of robots. *Proc. IEEE Int. Conference on Robotics and Automation*, pp. 4215–4221, 2003.

[15] Y. Li, K. Gupta, S. Payandeh. Motion planning of multiple agents in virtual environments using coordination graphs. *Proc. IEEE Int. Conference on Robotics and Automation*, pp. 378–383, 2005.

[16] B. Natarajan. On Planning Assemblies. *Proc. Symposium on Computational Geometry*, pp. 299–308, 1988.

[17] B. Nnaji. *Theory of Automatic Robot Assembly and Programming*. Chapman & Hall, 1992.

[18] P. O'Donnell, T. Lozano-Perez. Deadlock-free and collision-free coordination of two robot manipulators. *Proc. IEEE Int. Conference on Robotics and Automation*, pp. 484–489, 1989.

[19] M. Peasgood, C. Clark, J. McPhee. A Complete and scalable strategy for coordinating multiple robots within roadmaps. *IEEE Trans. on Robotics* 24(2), pp. 283–292, 2008

[20] J. Peng, S. Akella. Coordinating multiple robots with kinodynamic constraints along specified paths. *Int. Journal of Robotics Research* 24(4), pp. 295–310, 2005.

[21] M. Saha, P. Isto. Multi-robot motion planning by incremental coordination. *Proc. IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, pp. 5960–5963, 2006.

[22] G. Sanchez, J.-C. Latombe. Using a PRM planner to compare centralized and decoupled planning for multi-robot systems. *Proc. IEEE Int. Conference on Robotics and Automation*, pp. 2112–2119, 2002.

[23] T. Siméon, S. Leroy, J.-P. Laumond. Path coordination for multiple mobile robots: a resolution complete algorithm. *IEEE Trans. on Robotics and Automation* 18(1), pp. 42–49, 2002.

[24] J. Snoeyink, J. Stolfi. Objects that cannot be taken apart with two hands. *Discrete and Computational Geometry* 12, pp. 367–384, 1994.

[25] S. Sundaram, I. Remmler, N. Amato. Disassembly sequencing using a motion planning approach. *Proc. IEEE Int. Conference on Robotics and Automation*, pp. 1475–1480, 2001.

[26] P. Švestka, M. Overmars. Coordinated path planning for multiple robots. *Robotics and Autonomous Systems* 23(3), pp. 125–152, 1998.

[27] J. van den Berg, M. Overmars. Prioritized motion planning for multiple robots. *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 2217–2222, 2005.

[28] R. Wilson, J.-C. Latombe. Geometric Reasoning About Assembly. *Artificial Intelligence* 71(2), 1994.

[29] K. Kant, S. Zucker. Towards efficient trajectory planning: The path-velocity decomposition. *Int. Journal of Robotics Research* 5(3), pp. 72–89, 1986.