Multi-Robot Grasp Planning for Sequential Assembly Operations

Mehmet Dogar and Andrew Spielberg and Daniela Rus

***Abstract*— We formulate multi-robot grasp planning for sequential assembly operations as a constraint satisfaction problem (CSP) and present an algorithm to solve it. Our algorithm, through the assumption of feasible regrasps, divides the CSP into independent smaller problems which can be solved very fast. The algorithm then improves this solution by removing increasing number of regrasps from the plan, yielding an anytime planner.**

I. INTRODUCTION

We are interested in multi-robot systems which can perform sequences of assembly operations to build complex structures. Each assembly operation in the sequence requires multiple robots to grasp multiple parts and bring them together in space in specific relative poses. Once an assembly operation is complete, the semi-assembled structure can be transferred to subsequent assembly operations to be combined with even more parts. We present an example in Fig. 1 where a team of robots assemble a chair by attaching parts to each other with fasteners.

This paper addresses the problem of finding robot configurations which grasp the parts during a sequence of assembly operations. We define a robot configuration as the complete pose of an individual robot. The problem imposes a variety of constraints on the robot configurations. Take the assembly operation scenes in Fig. 1. We immediately see one type of constraint: the robot bodies must not intersect and must avoid collision. In effect, they must "share" the free space.

The sequential nature of the task, however, may result in even more constraints. A robot may choose one of two strategies to move a semi-assembled structure from one assembly operation to the next (Fig. 1): The robot can *regrasp*, changing its grasp on the semi-assembled structure, or the robot can *transfer* the semi-assembled structure directly to the next operation, keeping the same grasp.

Both strategies have their advantages. If the robot chooses transfer, it avoids extra regrasp operations during execution. Regrasps, on the other hand, make the planning problem easier by decoupling sequence of operations from each other: In Fig. 1, since the robot commits to transfer the structure between assembly operations 1 & 2, it must plan a grasp of the part which works for both operations. The coupling between multiple operations makes it extremely expensive to solve problems with long sequences of assembly operations

Humans use a combination of both strategies during manipulation: we regrasp when we need to, but we are also able to use transfer grasps which work for more than one operation. Given a sequence of assembly operations, how can a team of robots decide when to regrasp and when to transfer? We present a planner with this capability: Our algorithm trades off between regrasps and transfers while generating collision-free robot configurations for each assembly operation.

We formulate multi-robot grasp planning as a constraint satisfaction problem (CSP). In this representation every robotic grasp in every assembly operation becomes a variable. Every variable must be assigned a robot configuration which grasps a particular part or semi-assembled structure. We impose two types of constraints: *collision constraints* between variables of the same assembly operation; and *transfer constraints* between variables in subsequent operations.

Ideally, a plan involves no regrasps and the assembly is transferred between operations smoothly. Trying to find a plan with no regrasps, however, means having transfer constraints between all operations. A complete solution requires solving for all the assembly operations at once. Complete CSP solvers display exponential complexity with respect to the number of variables. Solving the multi-robot grasp planning problem then becomes very expensive very fast with increasing number of assembly operations.

Instead, our algorithm starts with a strategy to perform regrasps between all operations. Our key assumption is that, regrasps between any two grasps (possibly through a series of intermediate grasps) are always feasible. This decouples assembly operations from each other. The resulting problem can be solved very fast by solving a small CSP separately for each assembly operation.

After finding this initial solution, our algorithm continues to find solutions with fewer regrasps by imposing sets of transfer constraints. As such solutions are found, the algorithm increases the number of transfer constraints imposed.

Our algorithm is an anytime planner: Given more time, it generates plans with fewer regrasps and more transfers.

When imposing a new set of transfer constraints, our algorithm does *not* solve the CSP from scratch: Solutions with fewer (or no) transfer constraints are readily available from previous cycles. We use state-of-the-art local search methods for CSPs, which are initialized with partial solutions. Local-search methods work only in a locality of the constraint graph and therefore their runtime is not affected by the full size of the CSP [1], leading to very fast updates.

A. Related work

Recent work by Lozano-Pérez and Kaelbling [2] represent manipulation problems as CSPs. These geometric CSPs are formulated by a higher-level task planner. Their focus is on

Computer Science and Artificial Intelligence Lab, MIT
{mdogar,aespielberg,rus}@csail.mit.edu

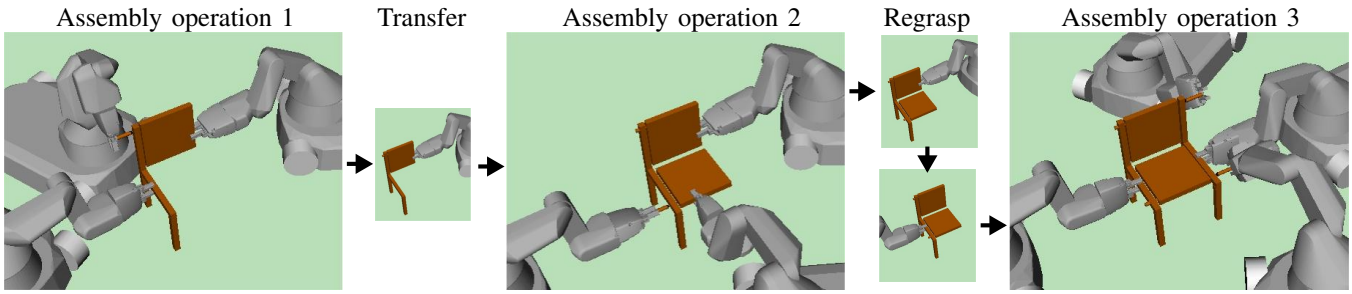

Fig. 1: Multi-robot assembly of a chair.

the interface between the task planner and CSP formulation, and they propose methods for constructing the CSPs efficiently. The CSPs are solved by an off-the-shelf solver. We propose an algorithm to solve the CSP itself by using domain-specific assumptions, such as feasible regrasps.

The effectiveness and necessity of regrasping during manipulation have been recognized [3, 4]. We show that assuming feasibility of regrasps we can simplify the CSP solutions of manipulation plans significantly. Structures similar to the *grasp-placement space* [5] or the *grasp-graph* [6] can be precomputed to satisfy our regrasp feasibility assumption.

Other grasp planners that take into account task constraints exist [7, 8, 9]. We focus on planning such grasps in a sequential and multi-robot context.

II. Problem

An assembly is a collection of simple parts at specific relative poses. Robots perform an *assembly operation*, $o = (\mathbf{A_{in}}, a_{out}, p)$, to produce an output assembly a_{out} from a set of input assemblies $\mathbf{A_{in}}$. We also assume that a three-dimensional pose in the environment, p, is specified as the location of an operation.

During an assembly operation, input assemblies $\mathbf{A_{in}}$ must be grasped and supported by robots at their respective poses in a_{out} at operation pose p. We assume that a local controller exists to perform the fastening/screwing, once the parts are at the poses specified by the assembly operation.

We use q to represent a *robot configuration*, which includes base pose and manipulator joint configurations. If a robot configuration q places the robot gripper at a grasping pose for assembly a during operation o, we say that "q is a grasping robot configuration for a during o".

Robots perform a sequence of assembly operations $\mathbf{O} = [o_i]_{i=1}^{N}$ to build large structures: output assemblies of earlier operations are used as inputs in later operations.

A. CSP Formulation

We formulate the problem of *multi-robot grasp planning for sequential assembly operations* as a CSP. A CSP is defined by a set of variables $\mathbf{X}$, a set of possible values $\mathbf{V}(x)$ that each variable x can be assigned with, and a set of constraints specifying consistent assignments of values to variables. A solution to the CSP is an assignment of values to all the variables that is consistent with all the constraints.

Variables: For our problem, we create one variable for the grasp of each input assembly of each assembly operation. We use ${}^{o}x^{a}$ to represent the variable correponding to the grasp of assembly $a \in \mathbf{A_{in}}$ of operation $o \in \mathbf{O}$.

Values: The set of values for the variable ${}^{o}x^{a}$ is the set of robot configurations grasping the assembly:

$$V({}^{o}x^{a}) = \{q \mid q \text{ is a grasping robot configuration for } a \text{ during } o.\}$$

In general there can be a continuous set of robot configurations grasping a. We discretize this continuous set by sampling uniformly at a fine resolution.

Constraints: We define two sets of constraints: *collision constraints* and *transfer constraints*. A collision constraint $c(x, x')$ enforces that two robot configurations x and x' do not collide. We create a collision constraint $c({}^{o}x^{a}, {}^{o}x^{a'})$ between each pair of variables of the same operation o.

A transfer constraint $t(x, x')$ enforces that robot configurations x and x' grasp the same part while placing the robot gripper at the same pose on the part. We can create a transfer constraint between variables of two consecutive operations. Suppose $o = (\mathbf{A_{in}}, a_{out}, p)$ and $o' = (\mathbf{A'_{in}}, a'_{out}, p')$ are consecutive operations such that $a_{out} \in \mathbf{A'_{in}}$; i.e. the output assembly of o is one of the input assemblies of o'. We can create a transfer constraint $t({}^{o}x^{a}, {}^{o'}x^{a_{out}})$ for any $a \in \mathbf{A_{in}}$.

If no transfer constraint with a previous operation exists for such an input assembly, then the robots will need to perform a regrasp.

III. Algorithm

Our algorithm (Alg. 1) first finds an "all-regrasps" plan, and then imposes transfer constraints.

We first assume no transfer constraints between operations. Collision constraints remain, but they only constrain variables within an operation. Hence, the constraint graph is divided into N connected components where N is the number of assembly operations.

We solve each of these connected components separately using a complete CSP solver (lines 2-4 in Alg. 1). We use backtracking search with forward-checking [1]. The combination of solutions for all operations give the overall "all-regrasps" solution (line 5).

Once the "all-regrasps" solution is found, our algorithm starts imposing a gradually increasing number of transfer constraints (line 6). On line 7, the function "NCombinationsOfTransConst(n)" generates a list of all valid n-combinations of transfer constraints. The algorithm

Algorithm 1 Multi-robot grasp planning for sequential assembly operations

Input: $\mathbf{O} = [o_i]_{i=1}^{N}$ is a sequence of assembly operations.

1: $\mathbf{X}, \mathbf{V} \leftarrow$ ComputeCSPVariablesAndValues($\mathbf{O}$)
2: **for each** o_i **in** $\mathbf{O}$ **do**
3: $\quad C[o_i] \leftarrow$ CollisionConstraints(o_i)
4: $\quad sol[o_i] \leftarrow$ SolveCSP($\mathbf{X}[o_i], \mathbf{V}[o_i], \mathbf{C}[o_i]$)
5: $best\_sol \leftarrow \{sol[o_i]\}_{i=1}^{N}$
6: **for** $n = 1$ **to** NMaxTransferConstraints **do**
7: $\quad$ **for each** $\mathbf{T}$ **in** NCombinationsOfTransConst(n) **do**
8: $\quad\quad sol \leftarrow$ SolveCSPLocal($\mathbf{X}, \mathbf{V}, \{\mathbf{C}, \mathbf{T}\}, best\_sol$)
9: $\quad\quad$ **if** sol **exists then**
10: $\quad\quad\quad best\_sol \leftarrow sol$
11: $\quad\quad\quad$ **break**

then loops over this list (line 7) trying to solve the new CSP by imposing the set of n transfer constraints together with the collision constraints (line 8). If a solution is found for a set of transfer constraints with size n, the solution is recorded as the new best solution, and the algorithm progresses to sets of size $n+1$ (lines 9-11). One can stop the algorithm anytime after the "all-regrasps" solution is found and use the current best solution.

We use local seach techniques for CSPs when we impose transfer constraints (line 8). Local techniques starts with an assignment of values to variables, identifies the conflict regions in the constraint graph, and tries to resolve the conflict only in the locality of these regions. Our algorithm re-uses the latest best solution to seed the local search. We use an implementation of the *min-conflicts* algorithm [1].

Our algorithm is complete: If there is a solution to the problem, our algorithm will at least return the "all-regrasps" solution, as we are using a complete CSP solver in the first part of our algorithm. Since local search methods, which we use in the second part of our algorithm, do not exhaust the search space, our algorithm may not be able to find the optimal solution; i.e. the solution with the minimum number of regrasps. Local search techniques, nevertheless, are known to display good performance in real world problems [1].

IV. Experiments and Results

We implemented and evaluated our algorithm on an example chair assembly operation. The number of robots required by the operations are 3, 3, and 4. The operations require the semi-assembled structures to be transferred two times: between operations 1-2, and 2-3. We implemented our algorithm and evaluated it in the OpenRAVE environment [10] with four KUKA YouBot robot models .

We ran our algorithm on the chair example 10 times. In 9 of these runs, our algorithm found the optimal solution with no regrasps (Fig. 3), and in 1 run it generated a solution with 1 regrasp and 1 transfer (Fig. 1). We plot the time it takes our algorithm to generate plans with different number of regrasps in Fig. 2. The horizontal bars show the standard deviations. Our algorithm generates the "all-regrasps" solution on average at 0.7 seconds and then gradually improves the solution on average every 0.1 seconds.

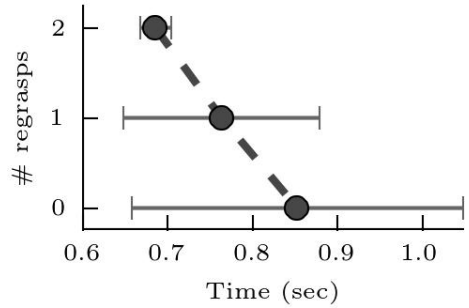

Fig. 2: Time to generate plans with decreasing number of regrasps.

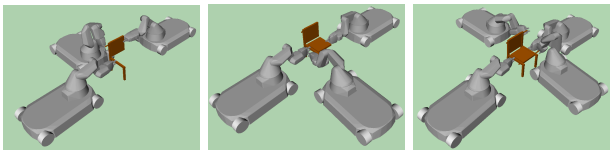

Fig. 3: Solution for the assembly of our chair example.

	Alg. 1	Naive optimal
Chair	0.86 (0.19)	255.36 (282.01)

TABLE I: Average planning times in seconds. Standard deviations are shown in parantheses.

We present an example optimal plan in Fig. 3. The left-most robot holding onto the side of the chair keeps its grasp fixed and transfers the semi-assembled structures between operations.

We also compare the performance of our algorithm with another algorithm which starts with the maximum number of transfer constraints it can impose, runs a complete CSP solver, and reduces the number of transfer constraints as solutions cannot be found. This algorithm is optimal, but also naive in that it tries to solve the full CSP at once. In Tab. I we present the planning times of our algorithm (Alg. 1) and the naive optimal algorithm. Our algorithm is two orders of magnitude faster.

References

[1] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed., 2003.
[2] T. Lozano-Pérez and L. P. Kaelbling, "A constraint-based method for solving sequential manipulation planning problems," in *IROS*, 2014.
[3] T. Lozano-Pérez, J. Jones, E. Mazer, P. O'Donnell, W. Grimson, P. Tournassoud, and A. Lanusse, "Handey: A robot system that recognizes, plans, and manipulates," in *ICRA*, 1987.
[4] T. Siméon, J.-P. Laumond, J. Cortés, and A. Sahbani, "Manipulation planning with probabilistic roadmaps," *IJRR*, vol. 23, no. 7-8, 2004.
[5] P. Tournassoud, T. Lozano-Pérez, and E. Mazer, "Regrasping," in *ICRA*, 1987.
[6] N. Dafle, A. Rodriguez, R. Paolini, B. Tang, S. Srinivasa, M. Erdmann, M. Mason, I. Lundberg, H. Staab, and T. Fuhlbrigge, "Extrinsic dexterity: In-hand manipulation with external forces," in *ICRA*, 2014.
[7] D. Berenson and S. S. Srinivasa, "Grasp synthesis in cluttered environments for dexterous hands," in *Humanoids*, 2008.
[8] D. Berenson, S. S. Srinivasa, and J. Kuffner, "Task space regions: A framework for pose-constrained manipulation planning," *IJRR*, 2011.
[9] H. Dang and P. K. Allen, "Semantic grasping: Planning robotic grasps functionally suitable for an object manipulation task," in *IROS*, 2012.
[10] R. Diankov, "Automated construction of robotic manipulation programs," Ph.D. dissertation, CMU, Robotics Institute, August 2010.