In Proceedings ICRA 2010, May 2010,Anchorage, Alaska.

Robust Robotic Assembly through Contingencies, Plan Repair and Re-Planning

Frederik W. Heger and Sanjiv Singh

***Abstract*— Enabling mobile robots to assemble large structures in constrained environments requires planning systems that are both capable of dealing with high complexity and can provide robust execution in the face of run-time failures. We achieve execution robustness through exception handling capabilities that are seamlessly integrated throughout the planning system. Having these recovery mechanisms in place allows us to leverage their capabilities to compensate for problems introduced by approximations made during planning. Turning an apparent problem into an opportunity, we are able to plan complex assembly tasks and execute them robustly without the computational cost associated with more sophisticated planners and apply some of the savings toward recovering from unforeseen run-time errors. We show results where simple planning strategies paired with exception-handling are able to achieve the same outcomes (and in less time) as more elaborate methods would.**

I. Introduction

The first reaction to the term "robotic assembly" is usually a mental picture of an industrial assembly line where stationary robots perform repetitive tasks at high speeds and with high precision. That is not the kind of "assembly" our work is about. Instead, we consider mobile manipulators retrieving components from a storage location, transporting them through their environment and assembling them into a large structure. We are developing a framework for planning assembly tasks that, given a desired goal structure, automatically decomposes the task and commands robots to execute them. As with all real-robot systems, things can (and will) go wrong during task execution. We leverage the availability of a skilled operator to provide exception-handling at all levels of the system – from low-level behaviors, to task-level execution to high-level (re-)planning. The result is a robotic assembly system capable of robustly executing complex tasks in constrained environments with implicit flexibility to adapt and modify the plan in response to run-time errors.

A. Motivation

With robotic manipulation increasing in capability and availability, a natural next step for mobile robots to advance beyond merely navigating through and sensing their surroundings is to actively modify their environment. Assembly of structures is an important application on its own, and it is a good example of a class of tasks that require complex coordination of multiple robots.

F. Heger is a Ph.D. student and S. Singh is a Research Professor at the Field Robotics Center, The Robotics Institute, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, U.S.A. {fwh,ssingh}@ri.cmu.edu

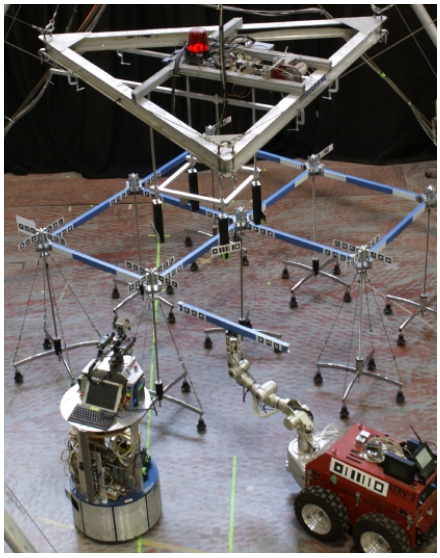

Fig. 1. A nearly completed lattice of 21 components. Three robots (a mobile manipulator, a dedicated sensing agent and a crane) cooperate to perform the assembly. Our system can plan for and execute the construction of any subset of this grid.

While multiple robots may have to tightly cooperate for certain parts of the assembly (e.g., where large components are involved), large portions of the work can usually be performed by multiple robots in parallel, working on separate parts of the structure largely independently. Since they are still all operating in a constrained environment, in order to ensure safety, robots either have to be well coordinated even when their tasks may not require it (i.e., motions planned in the joint configuration space of all robots, as well as with time dependence), or there have to be other mechanisms in place to avoid collisions and impasses that arise.

Interesting structures are comprised of numerous components that have to be brought together and connected into a growing obstacle for the robots performing the work. The complexity of the problem quickly reaches the limits of current planning techniques. At the same time, for a robotic assembly system to be useful in real-world scenarios, it has to be robust to execution-time failures.

We see the inevitability of failures occurring during execution not as a problem, but an opportunity. Instead of expending much effort on using highly sophisticated planning methods (which still need another level of error handling to provide the desired robustness of the overall system), we use simpler (and faster to compute) techniques that produce solutions for many nominal situations and equip the system with powerful recovery mechanisms to resolve problems as they arise.

B. Overview

Our work extends the traditional three-tier architecture (planning, executive, behaviors, [1]) by seamlessly integrating recovery mechanisms for execution-time exceptions at all levels of the hierarchy (earlier efforts focused exclusively on error recovery at the lowest behavioral level). This paper investigates how these comprehensive strategies for error recovery can be employed to ensure robust robot performance even as strong simplifying approximations are made to improve planning efficiency. The main idea is that since error recovery methods are essential for robust operation in any system, they can be leveraged to compensate for any discrepancies between the planner's (simplified and approximated) representation and the real world.

After reviewing related work, we present the details of our approach that enable robust robotic assembly by incorporating powerful exception handling mechanisms throughout the system. We show results of an experiment to test the effectiveness of our approach in simulated assembly scenarios and discuss our findings before concluding and pointing out directions of future work.

II. Related Work

Prior work relevant to assembly planning falls into two main categories: approaches that treat assembly as a sequencing problem on an abstract symbolic level, and ones that consider fine-grained motions of the robots involved. We see both as integral aspects of a larger problem that cannot be solved well with either method alone.

1) Symbolic Planning: Symbolic methods abstract problems into simple operators with preconditions and effects. A sequence of operators that transform given initial conditions into desired final configurations describes a plan for the scenario [2], [3]. Such approaches efficiently exploit the step-by-step nature of many problems by abstracting away difficult to compute constraints into simple heuristics. This abstraction, however, limits the reasoning about the real world to queries that have to be answered by an outside process (e.g., an oracle). Infeasibility of a plan often cannot be detected until the robots – during execution – come to a dead end because a workspace constraint was unknown during planning. Plan verification systems can help reduce such problems by verifying a symbolic plan step by step either after it is completed or while it is being planned [4].

2) Motion Planning: Other applicable work focuses on the motion planning aspects of the problem [5], [6], [7], but plans produced by those approaches do not always satisfy critical structural constraints imposed by the structure to be assembled. Koditschek et al.'s navigation functions, for example, assumes that all states between the initial and final configurations are valid (including partially assembled components), which is not the case in assembly scenarios. There is no guarantee that extrema in the navigation functions where assembly roles change coincide with valid assembly states where such a change is allowed.

In the context of this paper, motion planners that are capable of planning for robots with complex motion models (i.e., non-holonomic motion constraints, etc.) are of particular interest [8]. Planning on a lattice that encodes the robots' motion capabilities, they can guarantee feasible plans, but they also are computationally expensive.

Since assembly scenarios have a distinct underlying step-by-step structure, pure motion planning approaches do not produce the results we are looking for. Stilman et al.'s navigation among movable obstacles [9] plans first in an abstract graph of configuration space segments and then uses motion planning techniques to evaluate paths suggested by graph edges. Manipulation planning is faced with similar challenges to assembly planning at a finer level of detail (e.g., dextrous motions to grasp and re-grasp components [10], [11]). An assembly plan sets up manipulation planning problems for each step in the plan.

3) Assembly Planning: Homem de Mello developed a representation for describing mechanical assembly sequences based on AND/OR graphs [12], [13] similar to our representation [14]. Using this graph structure, he presented a complete and correct algorithm for generating assembly sequences of a desired configuration by planning the disassembly of the goal structure [15]. Existing approaches are limited to highly structured environments (e.g., work cells, [12], [4]) and are usually concerned with assembly feasibility and serviceability of parts in assemblies. The focus is on optimal plans to maximize efficiency of the assembly/production process. Once a plan has been found, it will be executed thousands of times without variation.

4) Robotic Assembly: In addition to our own work in multi-robot assembly [1] we are aware of one other group where real robots cooperate to assemble a (simple) structure [16]. Both efforts thus far focus on the execution part of the problem and operate according to a simple script written by hand that is followed by the robots. The planning system we describe here will replace manual scripting of assembly actions with automatic tasking of assembly robots based on a high-level goal specification of the structure to be assembled.

III. Approach

For robotic assembly systems to be useful in real-world applications, they must be able to operate robustly, meaning the desired structure must be assembled, even if exceptions occur along the way. We split the overall assembly planning process into three stages: a first (offline) pass that considers only structural constraints, a second (also offline) pass where available robots and their capabilities are considered, and a final (online) process to adapt and modify the plan to changing conditions and exceptions during execution.

Our approach is to use simple planning strategies (that are efficient to compute and yield good plans within their limited representation), and then provide exception handling mechanisms (both autonomous ones and expert-operator-based ones) to repair and re-plan tasks as execution-time exceptions occur. Fig. 2 provides an overview of our robotic assembly system. Given as input a desired structure to be assembled by mobile manipulators in their environment, the assembly planner represents the problem as an assembly

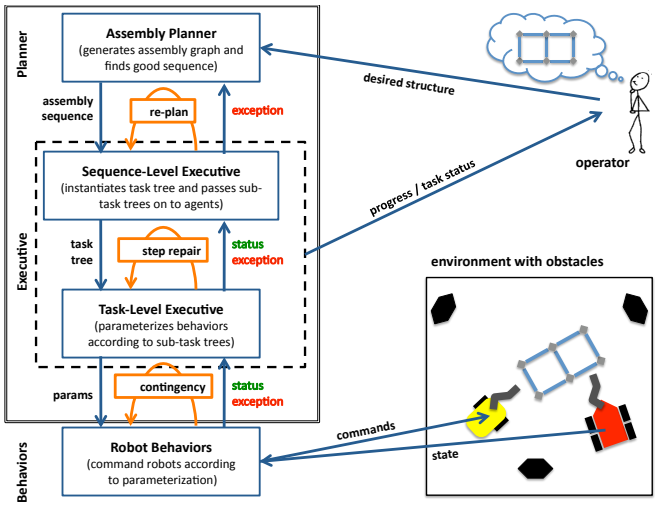

Fig. 2. A high-level overview of the robotic assembly system. Exception handling capabilities are seamlessly integrated throughout the system to recover from unforeseen errors where it is most appropriate.

graph that it searches for the best assembly plan. As the quality of a plan directly depends on robots' motions through the workspace, motion planning techniques are required to evaluate different assembly steps. Once an assembly plan is found, the executive is responsible for tasking robots and parameterizing behaviors. As errors occur, they escalate up through the system until they can be resolved and execution can continue.

The key to our approach is that we are specifically using motion planning techniques that make simplifying assumptions. The idea is that the simple strategies are often sufficient, and where they are not, the exception handling mechanisms are powerful enough to maintain overall system robustness. In contrast, more elaborate planners often will provide only marginally better solutions in significantly longer computation time, and they also cannot entirely avoid execution-time exceptions.

We have presented our representation of assembly planning problems and the general process of plan generation in previous work [14]. For the purpose of this paper, we focus specifically on the underlying planning methods used, and how error recovery strategies tie back in with the overall representation.

A. *Planning with Approximations*

Assembly planning, even for small structures of up to 20 components, requires many (as many as 600,000 for a structure of 21 components) motion and manipulation planning problems to be solved to find a feasible and desirable assembly sequence. While the bulk of this computation occurs offline prior to execution, we still want to keep planning time to a reasonable amount, especially as the number of robots involved in assembly steps grows.

The correct way of planning motions for several robots operating in a constrained environment to build a structure would be to plan in the joint configuration space of all robots involved. Given the ever-changing geometry of the environment and the robots carrying components, this approach soon reaches the limits of tractability as the number of robots increases. In addition, unless time dependence and delays caused by runtime failures are specifically taken into account, any break in synchronization between the robots can cause problems requiring potentially expensive re-planning.

The key to our approach is to recognize that even though there can be multiple robots involved in the assembly, most of the time they are working on separate tasks in the same environment, and many of their transfer motions lead through mostly open space until they get close to the structure.

Our first simplifying approximation is to plan motions for individual robots, only considering the structure obstacle while ignoring other robots in the environment. Clearly, left at that, this is a recipe for disaster where robots will pile up in the center of the workspace as their paths intersect.

The second simplification comes from using simple motion models for the robots during planning (e.g., plan for a holonomic robot, even though the actual robot is skid-steered) instead of more accurate models that are more expensive to plan with. As before, robot motions between the storage location and the structure that mostly lead through free space are minimally affected by this discrepancy. As the robots get closer to the structure, problems will start to arise that need to be addressed.

In our system, motion plans are generated using a PRM-based motion planner from Stanford's MPK package [17] which allows us to easily specify the changing environment, as well as robots traveling alone or while carrying components. While the planner can handle several robots at once (in their combined configuration space), the required planning time grows so quickly that in practice, as we will show, planning for individual robots and dealing with the consequences works very well.

Our solution to the problems created by the two simplifications mentioned above is to rely on the same failure recovery techniques already in place to ensure execution robustness. The robots are able to detect exceptions (in this case: blocked paths), stop safely and request help (either from an autonomous repair/re-plan system or from a human operator). Depending on the situation, the recovery response can be a simple "continue now" (if the cause of the exception has passed, e.g., another robot temporarily in the way), a new motion plan from the current location (again with the same simplifications), or a larger-scale re-plan.

Figs. 3, 4 and 5 illustrate examples of how the approximate planning approach can lead to problems, and how those problems can be addressed by our system. Consider first a single robot. If its motion is planned assuming holonomic motion capabilities, but the robot is a skid-steered vehicle, it will attempt to "cut corners" when driving around obstacles (Fig. 3). If the robot stops before a collision occurs and generates a new path from there to its goal, it may get lucky and find a viable solution (still assuming holonomic motion) simply by starting from a different location in the environment. In practice, however, it is more likely that the new path will send the robot right back into the same obstacle. In this case, additional help is required to get the

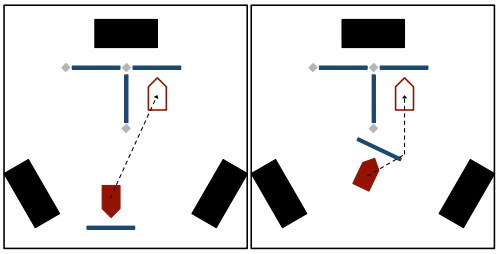

Fig. 3. A path planned assuming holonomic motion can cause a non-holonomic robot to "cut corners". Planning a new path from the failure location can sometimes solve the problem. Other times the new plan leads right back into the same problem, and more help is required.

robot unstuck. In our system, this request for help would be handled by the operator.

Planning paths for multiple robots operating in parallel on an individual robot basis leads to the obvious problem of paths crossing (Fig. 4). If only one of the robots is affected by the crossing (i.e., detects an imminent collision and stops to avoid it), it can usually resolve the exception by yielding to the other robot and then continuing once the original path is clear. If, however, both robots block each other, more assistance is needed. In this case, the operator is alerted and can decide how the robots should proceed (e.g., by backing up one robot, allowing the other to proceed, and finally releasing the first to follow its path).

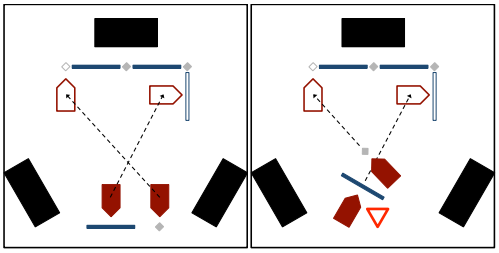

Fig. 4. If robots do not consider other robots in the environment, their paths are likely to cross at times. In the example shown, one robot yields and then can continue once the original path is clear. If the robots are blocking each other, additional help is required.

Fig. 5 shows a scenario where one robot's approach to an installation site is blocked by another robot (that was not considered during planning). Instead of waiting for the other robot to complete its task and move out of the way, the system (autonomously or with the help of the operator) can make slight modifications to a robot's immediate task to complete it in an alternate way. In the example scenario, the beam is installed from the right after the approach from the left as originally planned was found to be blocked.

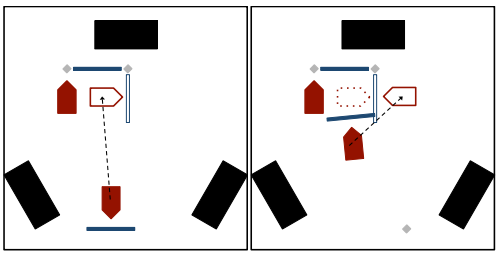

Fig. 5. If one robot's position keeps another one from reaching its desired target, the immediate assembly step may be repaired to achieve the original goal in a slightly different way.

All the problems we mentioned that are caused by using strong approximations during planning can be compensated for by recovery mechanisms built into the system.

B. Execution and Exception Handling

We consider three levels of failure recovery (see Fig. 2):

1) Contingencies: At the lowest level, as a first recovery attempt, each behavior has simple contingency responses for things that are known to go wrong from time to time. Often "try again" is a valid recovery strategy, or the operator can take control via teleoperation. For such contingency recovery actions, the assembly planner never gets involved. After a number of contingency attempts fail, more work is required to continue on with the task.

2) Assembly Step Repair: Each failed assembly operation is associated with an edge in the assembly graph. As a first attempt of recovery at the level of the assembly planner, we consider the failed graph edge and attempt to repair this particular step (Fig. 6 (left)). Enforcing the same final condition as in the original plan and taking into account any new information available due to the failure, the planner checks to see if there are alternative parameterizations of the failed task that allow it to repair the plan and then continue on as originally planned. Depending on how far along the assembly step the error occurred, the planner may have different (or none at all) options available to repair a step. If a repair is possible, the affected assembly step is reparameterized and execution continues (until more errors require further repairs).

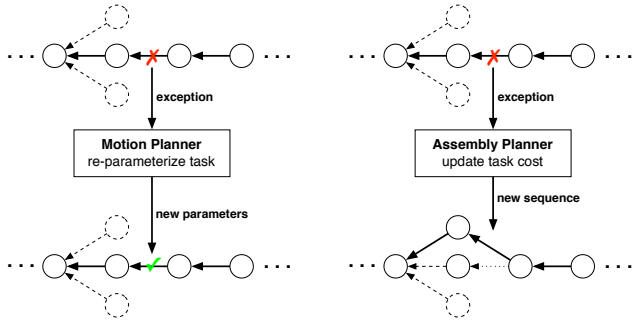

Fig. 6. Plan repair and re-planning. As execution-time failures occur, they are seamlessly handled at the appropriate level of the system hierarchy.

3) Assembly Sequence Re-Plan: If no repair is possible (either because the task had already progressed too far to allow for alternative parameterizations while still enforcing the required final condition, or because there is no other way to perform this particular step at the current point in the overall sequence), the exception jumps up to a higher level in the executive, and the planner is queried for a new sequence from the current state of the assembly to the desired goal state. In this case, the offending graph edge is marked impassable, and a new graph search is run from the source state of the failed edge to the original target state (Fig. 6 (right)). Note however, that the failed assembly step left the robot somewhere along its task, possibly carrying a component that it is trying to install. Thus, the re-planned sequence needs to be prepended with setup tasks that return the robots to a clean state from where to continue on with

the new plan. In our case, carried components are returned to their storage location and the braced structure is released before the new plan can be followed.

IV. Experiment

We conducted an experiment to determine the effects of trading off sophisticated planning techniques (in the interest of planning time) for a comprehensive exception handling system (in the interest of robustness) that patches any problems caused by the optimistic motion planner. We noticed that as the number of exceptions increase (not unexpectedly) if the robot's motion is more complicated than the planner's assumptions, execution robustness can still be achieved with only a small number of directed instances of operator assistance that get the robot back on track.

A. Setup

Using our simulated assembly environment, we planned the assembly of a two-square structure (a 13-component substructure of the lattice shown in Fig. 1) at two different goal locations (Fig. 7). We considered two different robots: a holonomic base that is able to move as the planner thought it could, and a skid-steer vehicle (planned for assuming holonomic motion capabilities).

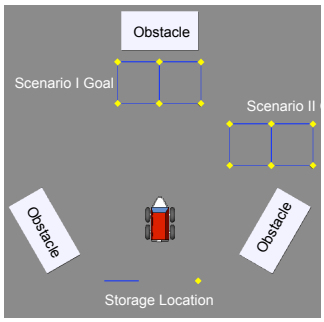

Fig. 7. The two experimental scenarios. The goal was to have the robot assemble a 13-component structure in constrained locations in the environment.

For each experimental condition, the planner produced an assembly sequence and then commanded a simulated mobile manipulator to execute it. During execution, a number of exceptions were triggered. If the robot detected an imminent collision of its body or an element it carried with another object, it stopped and threw a "Clearance" exception (this is caused by the "corner cutting" problem mentioned above). In addition, "Sensing" and "Manipulation" exceptions were generated randomly. For this experiment, 20% of all alignments suffered "Sensing" exceptions (i.e., in the real world, the robot would be unable to sense everything it needs to align itself with its target), and 40% of all manipulation attempts would fail (i.e., the robot would try to pick up a component, but something goes wrong).

"Clearance" exceptions trigger a re-plan of the robot's motion from its position where the failure occurred to the current goal position. "Sensing" and "Manipulation" exceptions are recovered from using a "Try Again" strategy. After five exceptions in a single assembly step, a more elaborate plan repair is triggered where the current step is re-parameterized from the current state to the step's goal state. If, after three repair attempts there is still no solution, the exception escalates to a re-planning strategy where the assembly graph is queried for an alternative sequence to the assembly goal state.

B. Results

For each scenario and experimental condition, we recorded the number of each type of exception that occurred, as well as the number of required operator interventions and the outcome of the run. Each experimental condition was run five times and the results averaged.

Mode	Clearance	Sensing	Manipulation	Op. Assistance	Repair	Re-Planning	Success
holonomic	0.0	20.8	16.6	0.0	3.0	0.0	100%
non-holonomic	6.4	14.6	17.4	1.6	2.4	0.0	100%

TABLE I

Scenario I: centered in workspace. The table shows the average number of each type of exception and recovery events, and the success rate of all runs.

In the less constrained structure goal position (Scenario I, Table I), the difference between the two experimental conditions (with the exception of "Clearance" exceptions and operator interventions) was very small. Since the holonomic robot moved the way the planner assumed it would, there were no instances of the robot trying to cut corners, and consequently, no operator assistance was required.

In Scenario II (Table II), in addition to the occurrence of "Clearance" exceptions and operator interventions for the realistic robot case, the system required more plan repair operations to complete the assembly, but as with all experimental conditions, all assembly runs finished successfully with the structure built as desired.

The repair instances where the system requested operator intervention could all be recovered by simply backing the robot up slightly or turning it away from an obstacle before the autonomous system could take control and continue on with task execution.

Mode	Clearance	Sensing	Manipulation	Op. Assistance	Repair	Re-Planning	Success
holonomic	0.0	15.8	11.4	0.0	2.2	0.8	100%
non-holonomic	15.6	22.8	17.8	4.2	5.4	1.0	100%

TABLE II

Scenario II: close to obstacle at edge of workspace. The table shows the average number of each type of exception and recovery events, and the success rate of all runs.

V. Discussion

In terms of overall system performance, the impact of using an optimistic (and simple to compute) motion planner to evaluate assembly steps during planning is limited to requiring some additional repair events, but the success rate is not affected. For the target applications of our system, increased execution time can be accommodated to achieve the required robust performance.

Of the "Clearance" exceptions due to differences between the optimistic planner and the less capable robot, some were recoverable autonomously while others required assistance from the operator. When the robots detected imminent collisions, in some cases the robot simply being in a different position in the workspace allowed the optimistic motion planner to generate a different enough recovery trajectory that the realistic robot was able to successfully follow around the obstacle. In many cases, however, the autonomous contingency recovery got stuck in a loop where (in the representation of the planner) the path was not really blocked, and thus the "recovery" trajectory was the same as the initial one, leading to an immediate re-failure. In those cases, the operator had to move the robot via teleoperation (the recovery action required less than 30 seconds) before the system could take over again.

Recovery and re-planning capabilities are necessary for the system to successfully perform assembly tasks even when using a motion planner that faithfully represents the capabilities of the motion planner to deal with unforeseen and unforeseeable exceptions. Since the complexity of the assembly representation already requires significant resources in terms of memory, and since during planning a large number of motion planning solutions need to be found to evaluate assembly steps, being able to use a simple yet optimistic motion planner augmented with an exception handling system that patches any problem during plan execution enables fast planning and robust execution.

VI. Conclusion

The exception handling capabilities already incorporated into our assembly planning system to ensure robust plan execution are also able to recover from errors caused by representational deficiencies and discrepancies between the motion planner's representation and the robot's true capabilities. We can exploit this already-present functionality to deliberately scale back the representational fidelity required of the planner we use to speed up planning without sacrificing overall system performance. This trade-off will allow us to work with larger problems than we could if high-fidelity planning were required at every step along the way.

The goal of assembly planning is to be able to work with large structures of many components. The inherent complexity makes this a challenging problem both from an efficiency and scalability point of view. We are investigate approaches incrementally searching the assembly graph where we can focus the search better to promising portions of the assembly graph, and maybe even avoid constructing the entire graph unless necessary for a particularly difficult structure. Scalability is concerned with large structures that can be broken up into smaller sub-structures, which can then become atomic parts in larger structures of structures. Multiple (teams of) robots and cooperative behaviors will also become increasingly important at that stage.

References

[1] B. Sellner, F. W. Heger, L. M. Hiatt, R. Simmons, and S. Singh, "Coordinated Multi-Agent Teams and Sliding Autonomy for Large-Scale Assembly," *Proceedings of the IEEE*, vol. 94, no. 7, July 2006.

[2] S. E. Fahlman, "A Planning System for Robot Construction Task," MIT Artificial Intelligence Laboratory, Tech. Rep. AITR-283, 1973.

[3] H. L. S. Younes and R. Simmons, "VHPOP: Versatile Heuristic Partial Order Planner," *Journal of Artificial Intelligence Research*, vol. 20, pp. 405–430, December 2003.

[4] S. G. Kaufman, R. H. Wilson, R. E. Jones, T. L. Calton, and A. L. Ames, "The Archimedes 2 Mechanical Assembly Planning System," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, vol. 4, Minneapolis, MN, April 1996, pp. 3361–3368.

[5] H. I. Bozma and D. E. Koditschek, "Assembly as a Noncooperative Game of its Pieces: Analysis of 1D Sphere Assemblies," *Robotica*, vol. 19, pp. 93–108, 2001.

[6] C. S. Karagöz, H. I. Bozma, and D. E. Koditschek, "Feedback-Based Event-Driven Parts Moving," *IEEE Transactions on Robotics*, vol. 20, no. 6, pp. 1012–1018, December 2004.

[7] J. Lengyel, M. Reichert, B. R. Donald, and D. P. Greenberg, "Real-Time Robot Motion Planning Using Rasterizing Computer Graphics Hardware," *Computer Graphics*, vol. 24, no. 4, pp. 327–335, 1990.

[8] M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Differentially Constrained Mobile Robot Motion Planning in State Lattices," *Journal of Field Robotics*, vol. 26, no. 3, pp. 308–333, March 2009.

[9] M. Stilman and J. J. Kuffner, "Navigation Among Movable Obstacles: Real-Time Reasoning in Complex Environments," in *Proceedings of the International Conference on Humanoid Robotics (Humanoids)*, 2004.

[10] C. L. Nielsen and L. E. Kavraki, "A Two Level Fuzzy PRM for Manipulation Planning," in *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, Takamatsu, Japan, November 2000.

[11] F. Gravot, R. Alami, and T. Siméon, "Playing with Several Roadmaps to Solve Manipulation Problems," in *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, Lausanne, Switzerland, October 2002.

[12] L. S. Homem de Mello and A. C. Sanderson, "Autommatic Generation of Mechanical Assembly Sequences," Carnegie Mellon University, The Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-88-19, December 1988.

[13] L. S. Homem de Mello, "Task Sequence Planning for Robotic Assembly," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, May 1989.

[14] F. W. Heger, "Generating Robust Assembly Plans in Constrained Environments," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, Pasadena, CA, May 2008.

[15] L. S. Homem de Mello and A. C. Sanderson, "A Correct and Complete Algorithm for the Generation of Mechanical Assembly Sequences," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 2, pp. 228–240, April 1991.

[16] A. Stroupe, T. Huntsberger, A. Okon, and H. Aghazarian, "Precision Manipulation With Cooperative Robots," in *Multi-Robot Systems: From Swarms to Intelligent Automata*, L. Parker, F. Schneider, and A. Schultz, Eds. Springer, 2005.

[17] G. Sanchez and J.-C. Latombe, "A Single-Query Bi-Directional Probabilistic Roadmap Planner with Lazy Collision Checking," in *Proceedings of the International Symposium on Robotics Research (ISRR)*, Lorne, Victoria, Australia, November 2001.