A Case for Collaborative Construction as Testbed for Cooperative Multi-Agent Planning*

Sven Koenig
Department of Computer Science
University of Southern California
skoenig@usc.edu

T. K. Satish Kumar
Department of Computer Science
University of Southern California
tkskwork@gmail.com

Abstract

Cooperative multi-agent planning is an understudied but important area of AI planning due to the increasing importance of multi-agent systems. In this paper, we make the case for using collaborative construction as testbed for cooperative multi-agent planning. Planning for single agents is already difficult due to the large number of blocks and long plans. Planning for multiple agents is even more difficult since it needs to reason about how to achieve a high degree of parallelism without agents obstructing each other even though many agents operate together in tight spaces. In previous research, we developed a first (domain-dependent, centralized and non-optimal) multi-agent planning method for this domain. Here, we explain the advantages of using collaborative construction as multi-agent planning domain, formalize the planning problem and relate it to existing planning problems in the hope that other researchers will adopt it as testbed for cooperative multi-agent planning.

Introduction

Cooperative multi-agent planning is an important area of AI planning due to the increasing importance of multi-agent systems. For example, teams of agents are more fault-tolerant and allow for more parallelism than single agents. Multi-agent planning promises to coordinate agents much more efficiently than alternative coordination strategies, such as — for example — behavior-based, stigmergy-based or market-based methods, which are typically very myopic. Yet, cooperative multi-agent planning is currently understudied. For example, only two out of 20 sessions at the International Conference on Automated Planning and Scheduling 2016 were on distributed and multi-agent planning.

In this paper, we make the case for using collaborative construction as testbed for cooperative multi-agent planning. Building on our previous research (Kumar, Jung, and Koenig 2014; Cai et al. 2016), we suggest to simulate the Harvard TERMES robots, actual robots that were inspired by termites. Among many other species of animals, termites are capable of building mounds that are much larger than themselves. Inspired by termites and their building activities, the Harvard TERMES project investigated how multiple robots can cooperate to build user-specified three-dimensional structures much larger than themselves (Petersen, Nagpal, and Werfel 2011). The agents need to build ramps to reach high places and avoid obstructing each other. Currently, the TERMES robots coordinate using local behavior-based and stigmergy-based rules, which make it impossible for them to construct complex structures. Planning is required, even for single agents, to build structures effectively since they need to build ramps to reach high places, for example when building towers. Ramps consist of many blocks and are time-consuming to build. Thus, agents need to plan carefully when and where to build ramps and, once built, how to utilize them best. Planning for single agents is already difficult due to the large number of blocks and long plans. Planning for multiple agents is even more difficult since it needs to reason about how to achieve a high degree of parallelism without agents obstructing each other even though many agents operate together in tight spaces. Furthermore, single agents no longer have to carry blocks all the way to their destinations since agents can hand over blocks to each other. For example, they can form bucket brigades to transport blocks. Ideally, one would like to plan for a large number of agents, such as one hundred agents or more.

Advantages of Collaborative Construction

Advantages of using collaborative construction as testbed for cooperative multi-agent planning include: The TERMES robots are very simple to simulate and allow for deterministic and symbolic planning. Robotics domains are often continuous domains with a substantial amount of sensor and actuator uncertainty, while AI planning is often studied in the context of symbolic domains without uncertainty.

*This paper re-uses a small amount of text from our previously published feasibility studies at ICAPS and AAMAS. We thank Marcello Cirillo, Tansel Uras and Liron Cohen for their suggestions and helpful discussions, Tansel Uras and Liron Cohen also for their help by experimenting with general-purpose planners in our domain, and Radhika Nagpal for commenting on a draft of this paper, her interest in our project and her encouragement. Our research was supported by NSF under grant numbers IIS- 1319966 and IIS-1409987 and ONR under grant number N00014-09-1-1031. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations, agencies or the U.S. government.

Collaborative construction fits the latter assumptions well. For example, the TERMES robots move on the blocks and the environment is thus automatically discretized into square cells. The TERMES robots use hardware features to achieve close-to-perfect execution. For example, a white cross on a black background of each block helps them to track both their position and orientation. Moreover, a circular indentation on each block helps them to turn in place without accumulating drift. Collaborative construction is thus a good domain for demonstrating the applicability of AI planning to robot planning, which will help to bridge the current gap in planning between AI and robotics. Collaborative construction pushes the state-of-the-art of AI planning but is not too difficult. There is a potential progression of research from centralized planning for single agents, via centralized planning for multiple agents to decentralized planning for multiple agents. At the same time, collaborative construction is also rich in structure that can be exploited for efficient and effective planning. In particular, spatial constraints, different from temporal constraints, are an understudied but very important area of AI planning due to the increasing importance of physical agents, such as robots, that operate in tight spaces. Collaborative construction subsumes some previously studied planning problems with spatial constraints. For example, parallelism comes with the overhead of having to coordinate multiple agents so that they neither collide with each other nor block each other. This introduces combinatorial problems akin to multi-agent path finding. Multi-agent path finding is concerned with multiple agents having to navigate effectively in tight spaces (Sharon et al. 2015; Wilt and Botea 2014), such as spaces with long narrow corridors where agents cannot pass each other. Multi-agent path finding is often studied in the context of automated warehouse domains, such as the Amazon fulfillment centers (Wurman, D'Andrea, and Mountz 2008), but is also relevant for collaborative construction since ramps are time-consuming to build and thus will typically be so narrow to not let two agents pass each other. Overall, we expect research on collaborative construction to yield insights into spatial multi-agent planning that go well beyond collaborative construction.

Hardware System

Our description of the TERMES hardware system follows (Petersen, Nagpal, and Werfel 2011; Werfel, Petersen, and Nagpal 2011; 2014), see Figure 1.[1] It consists of small autonomous mobile robots and a reservoir of passive "building blocks," simply referred to as "blocks." The robots gather blocks from the reservoir to collaboratively build a user-specified structure. The robots are roughly of the same size as the blocks. Yet, they can manipulate these blocks to build structures that are much larger and taller than themselves. They do so by stacking the blocks onto each other and building ramps to scale to greater heights.

The robots are equipped with four small whegs that allow for different kinds of locomotion using the same action of simply driving forward. The whegs allow the robots to move on level ground, climb up one block (to reach higher levels) and climb down one block (to reach lower levels) without any additional hardware or software capabilities, making this a reliable operation without complicated low-level control and allowing one to focus on high-level planning. The robots can navigate on a partially built structure (or the ground) without losing track of where they are or falling down. They are equipped with an arm and a gripper to facilitate picking-up, carrying and dropping-off blocks, one at a time. Mechanical features of the blocks also help the robots to perform these operations reliably with the use of only one actuator. One case where actuation is often not sufficiently accurate, which we ignore in the formalization below, is that it is difficult for robots to drop off a block directly between two other blocks (Petersen, Nagpal, and Werfel 2011).

[1] See www.eecs.harvard.edu/ssr/projects/cons/termes.html for more information.

Formalization of Collective Construction

We are given a start configuration and a desired goal configuration in form of user-specified 2D matrices of non-negative integers, referred to as workspace matrices. The cells of the matrices represent physical locations on a grid frame of reference, and the non-negative integers represent the heights of the towers (that is, vertical stacks of blocks) that need to be constructed by stacking blocks at those cells. (A height of zero means no tower, represented by a missing number in our figures.) Thus, the configurations do not have blocks that rest only partially on top of other blocks nor completely enclosed spaces, such as rooms in a house. As an example, consider an empty start configuration and a goal configuration that represents a castle that consists of a tower of height three surrounded by a wall of height one, as shown in Figure 2(d).

At any intermediate stage, the top of a tower is called *reachable* if and only if an agent, starting from the ground level, can reach the top of that tower by repeatedly turning left, turning right and driving forward. Turning left and right turns the agent 90 degrees in place. Moving the agent forward moves it to the neighboring tower in front of it as long as the agent moves at most one block up or down. Each agent can carry at most one block. The agent can pick up a block from the top of a tower if and only if there is a neighboring tower (in one of the four main compass directions) of height one less, the top of which is reachable, because it can then move to this neighboring tower and pick up the block. The agent can drop off a block on top of a tower if and only if there is a neighboring tower of equal height, the top of which is reachable, because the agent can then move to this neighboring tower and drop off the block. The problem is to build a user-specified structure with a given number of agents.

State-of-the-Art of Collective Construction

Currently, the TERMES robots coordinate using local behavior-based and stigmergy-based rules (Petersen, Nagpal, and Werfel 2011), which make it impossible for them to construct complex structures. There has

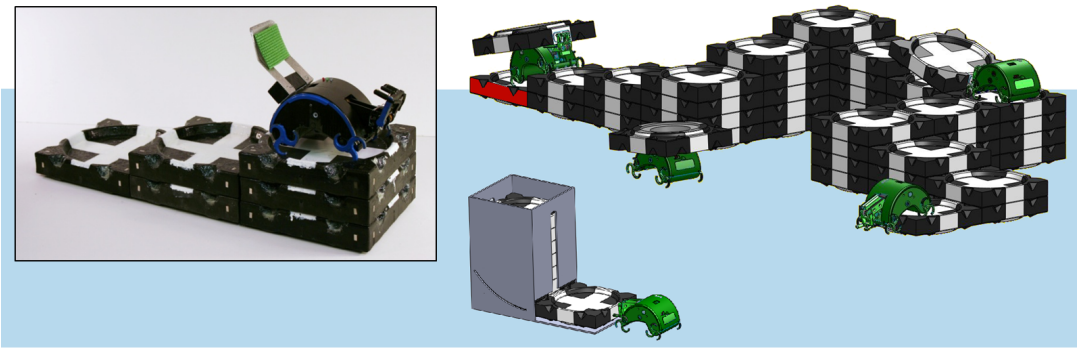

Figure 1: The TERMES system consisting of robots, blocks and the reservoir [figure courtesy of (Petersen, Nagpal, and Werfel 2011)].

been a fair amount of theoretical work on collective construction with different assumptions, including – but not limited to – (Jones and Mataric 2004; Grushin and Reggia 2008; Napp and Klavins 2010; Yun and Rus 2010). Many of these models are abstractions that do not model real hardware and often also make additional simplifications to allow either for a theoretical analysis or the development of simple planning methods.

Existing Planning Problems

The collective construction problem has similarities to the blocksworld and logistics problems, both of which have extensively been used in planning competitions (McDermott 2000). The blocksworld problem is concerned with building towers of blocks. The constraints are due to the vertical arrangements of blocks since only the top block of each tower can be accessed. How the blocks are moved does not impose any constraints, except that only a given number of blocks can be moved at the same time. The logistics problem, on the other hand, is concerned with moving objects to given cities. The constraints are due to how objects can be moved from city to city, given by the transportation options and their capacities. The spatial arrangement of objects does not impose any constraints. Planning researchers have also studied combinations of the blocksworld and logistics domains (Ghallab, Nau, and Traverso 2016) where both the spatial arrangement of objects and how they are moved imposes constraints, for example, in the context of moving containers in container terminals.

Collaborative construction imposes more constraints than the blocksworld, logistics or container terminal problems since blocks can be picked up from the tops of towers only when certain spatial conditions hold, blocks can be put down on tops of towers only when certain spatial conditions hold, and they need to be carried from their pick-up locations to their drop-off locations along spatially-feasible paths. This presents a variety of issues not present in the existing planning problems. For example, ramps of many blocks need to be built to satisfy the spatial conditions, resulting in long plans with many objects — which makes planning very time consuming already for a single agent. Collective construction with multiple agents is even more difficult since one needs to figure out how to achieve a high degree of parallelism even though the state space grows exponentially in the number of agents and multiple agents can easily obstruct each other in tight spaces.

Feasibility Study

No planning methods existed for collaborative construction with agents that correspond to the Harvard TERMES robots. We therefore considered it important to develop a planning method for this domain in a feasibility study before advocating its use as testbed for cooperative multi-agent planning. We assume in the following for simplicity that the reservoir (that provides the blocks) is unlimited and that the start configuration is empty, that is, all blocks are initially stored in the reservoir.

One intuitive single-agent planning method, called the tower-by-tower planning method, is to build the towers one by one, starting from one of the corners, each time constructing a ramp and then deconstructing it again. In this approach, the agent needs to build a ramp consisting of towers of heights $h-1, h-2 \ldots 1$ to build a tower of height h. The ramp is then deconstructed, resulting in $O(h^2)$ total number of block (pick-up and drop-off) operations to build a tower of height h. This intuitive planning method is correct, is complete, runs in polynomial time and performs a polynomial number of block operations for any user-specified structure. It demonstrates that any structure can be built by one or more agents provided that there is sufficient empty space around it. However, the tower-by-tower planning method is not very effective even for simple structures.

We therefore first developed a better single-agent planning method for this domain in a feasibility study. This planning method attempts to minimize the total number of block operations but is heuristic in nature, that is, is not guaranteed to achieve its objective (Kumar, Jung, and Koenig 2014). It is based on the idea of performing dynamic programming on a tree that spans the cells of the workspace

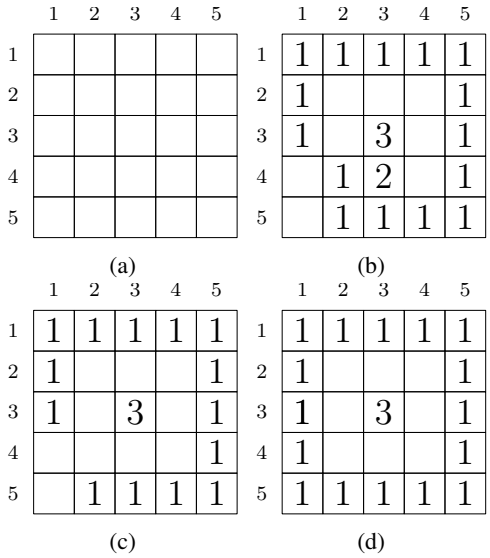

Figure 2: Shows an example of the phases of our planning algorithm. (a) shows the empty workspace matrix, which is the start configuration. (b) shows the workspace matrix after Phase 1, which adds blocks. A ramp for the tower and the tower itself have now been completely constructed, while the wall has been partially constructed. (c) shows the workspace matrix after Phase 2, which removes blocks. The ramp has now been deconstructed. (d) shows the workspace matrix after Phase 3 (goal configuration), which adds blocks. The remainder of the wall has now been constructed, finishing the goal configuration.

matrix and restricts the movements of the agent to the edges of this tree. The use of dynamic programming allows us to exploit common substructures and keep the number of block operations small. We then generalized it to a centralized multi-agent planning method (Cai et al. 2016). In the remainder of this section, we discuss both our single-agent planning method and its generalization.

Both planning methods operate on an undirected graph constructed from the workspace matrix. Each vertex corresponds to a cell, and each edge connects neighboring cells in the four compass directions. A special vertex S represents the reservoir and is connected to those vertices whose cells are neighbors of the reservoir, for example, the vertices whose cells are the boundary cells of the workspace matrix (since an agent carrying a block to or from the reservoir must cross the boundary of the workspace). The agents move on a spanning tree of this graph, rooted at S, as shown in Figure 3. They build a user-specified structure in phases. They add blocks to the structure in odd phases and remove blocks from the structure in even phases.

Our single-agent planning method uses dynamic programming on the spanning tree from its leaves to the root to first decide on the heights of the towers in each cell and then on the movements of the agent between the reservoir and the cells where blocks need to be added or removed. Consider, for example, a vertex in the spanning tree with two children

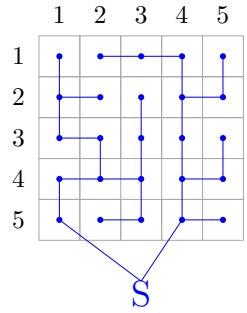

Figure 3: Shows the spanning tree used for the example.

during a phase where blocks are added to the structure. If dynamic programming has decided that towers of heights three and five are needed in the cells that correspond to the first and second child (respectively) of the vertex, then it can deduce that a tower of height four is needed in the cell that corresponds to the vertex itself since this height is necessary for an agent to put down the block that builds the tower of height five in the cell that corresponds to the second child. Details are given in (Kumar, Jung, and Koenig 2014), and an example is shown in Figure 2.

Our single-agent planning method is correct, is complete, runs in polynomial time and performs a polynomial number of block operations for any user-specified structure. It yields, within seconds, construction plans for problem instances with many blocks that require only 50-60 percent of block operations compared to the tower-by-tower planning method. However, our planning method can be improved. For example, its plan quality crucially depends on the spanning tree since the movements of the agent are restricted to the edges of the spanning tree, which can result in longer paths for the agent than necessary. One can find a good spanning tree for a given goal configuration in at least two ways: First, one can use heuristics for choosing good spanning trees. For example, minimum spanning trees on edge-weighted graphs can reduce the total number of block operations if the weight of an edge is the absolute value of the difference between the heights of the cells in the goal configuration that corresponds to the vertices it connects. (All edges neighboring S have weight zero.) Intuitively, a minimum spanning tree for this edge-weighted graph finds paths with minimum height variations in the goal configuration. Second, local search methods in an outer loop of our planning method can improve the initial spanning tree further over several iterations. Details are given in (Cai et al. 2016).

Our single-agent planning method can easily be generalized to multi-agent planning. Spanning trees allow for some parallelism in movements since the block operations carried out in the cells that correspond to the vertices of one subtree do not affect the operations for another subtree. For agents moving outside of their subtrees, one can exploit that multi-agent path finding is easier on trees and with identical agents (Yu and LaValle 2012). One simple technique to simplify coordination and avoid deadlocks is, for example, to move the agents in phases. All agents move along the tree

from the reservoir during odd phases and to the reservoir during even phases. Details are given in (Cai et al. 2016).

Conclusions

In this paper, we made a case for using collaborative construction with agents that correspond to the Harvard TERMES robots as testbed for cooperative multi-agent planning.

We have developed a first multi-agent planning method for this domain in a feasibility study. While elegant, it has a number of disadvantages. For example, it is domain-dependent, centralized, non-optimal, attempts to minimize the total number of block operations (rather than, say, the makespan, which is a more natural objective) and restricts the movements of the agents to the edges of the spanning tree, which can result in longer paths for the agents than necessary. The spanning tree also limits the amount of achievable parallelism for multiple agents. Agents can be assigned to different subtrees and then operate independently within their subtrees but still need to coordinate outside of their subtrees to avoid obstructing each other, for example, when picking up blocks from the reservoir. Finally, single agents do not have to carry blocks all the way to their destinations since agents can hand over blocks to each other, which our multi-agent planning method does not consider. Domain-independent, distributed and/or optimal planning raises a number of additional research issues that have not been addressed yet.

A student project at Ben-Gurion University of the Negev (Israel) recently encoded collaborative construction with the TERMES robots in a MA-PDDL format (Kovacs 2012) and used two existing (domain-independent) MA-STRIPS planners from the 2015 CoDMAP competition on toy instances that are much smaller than those that our (domain-dependent) planning method has been applied to (Yogev and Segal 2016). We predict that planning methods can be developed that are more efficient and effective than the ones existing one so far. Many different domain-dependent and domain-independent planning methods with different advantages and disadvantages are imaginable, such as learning a "ramp construction" macro to shorten the plan lengths and make planning more efficient. Smart macros are needed in this case since a crucial component of planning is to figure out how to fit ramps into the available space and how to amortize their construction and deconstruction effort among several towers that need to be built.

References

Cai, T.; Zhang, D.; Kumar, S.; Koenig, S.; and Ayanian, N. 2016. Local search on trees and a framework for automated construction using multiple identical robots [short paper]. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 1301–1302.

Ghallab, M.; Nau, D.; and Traverso, P. 2016. *Automated Planning and Acting*. Cambridge University Press.

Grushin, A., and Reggia, J. 2008. Automated design of distributed control rules for the self-assembly of prespecified artificial structures. *Robotics and Autonomous Systems* 56(4):334–359.

Jones, C., and Mataric, M. 2004. Automatic synthesis of communication-based coordinated multi-robot systems. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 381–387.

Kovacs, D. 2012. Multi-agent extension of PDDL3.1. In *Proceedings of the ICAPS Workshop on the International Planning Competition*, 19–27.

Kumar, S.; Jung, S.; and Koenig, S. 2014. A tree-based algorithm for construction robots. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*.

McDermott, D. 2000. The 1998 AI planning systems competition. *Artificial Intelligence Magazine* 21(2):35–55.

Napp, N., and Klavins, E. 2010. Robust by composition: Programs for multi-robot systems. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2459–2466.

Petersen, K.; Nagpal, R.; and Werfel, J. 2011. TERMES: An autonomous robotic system for three-dimensional collective construction. In *Proceedings of the International Conference on Robotics: Science and Systems (RSS)*.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. 2015. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence* 219:40–66.

Werfel, J.; Petersen, K.; and Nagpal, R. 2011. Distributed multi-robot algorithms for the TERMES 3D collective construction system. In *Proceedings of the Workshop on Reconfigurable Modular Robotics at the IEEE International Conference on Intelligent Robots and Systems (IROS)*.

Werfel, J.; Petersen, K.; and Nagpal, R. 2014. Designing collective behavior in a termite-inspired robot construction team. *Science* 343(6172).

Wilt, C., and Botea, A. 2014. Spatially distributed multiagent path planning. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*.

Wurman, P.; D'Andrea, R.; and Mountz, M. 2008. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI Magazine* 29(1):9–20.

Yogev, E., and Segal, A. 2016. A new problem domain for Classical MAS: 3D construction. Technical report, Department of Information System Engineering, Ben-Gurion University of the Negev. Supervisor: R. Stern.

Yu, J., and LaValle, S. 2012. Multi-agent path planning and network flow. In *Proceedings of the Workshop on Algorithmic Foundations of Robotics (WAFR)*, 157–173.

Yun, S., and Rus, D. 2010. Adaptation to robot failures and shape change in decentralized construction. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2451–2458.