Local Search on Trees and a Framework for Automated Construction Using Multiple Identical Robots*

(Extended Abstract)

Trevor Cai
Computer Science Dept.
Univ. of Southern California
tcai@usc.edu

David Y. Zhang
Google
New York City
dzhang21@gmail.com

T. K. Satish Kumar
Computer Science Dept.
Univ. of Southern California
tkskwork@gmail.com

Sven Koenig
Computer Science Dept.
Univ. of Southern California
skoenig@usc.edu

Nora Ayanian
Computer Science Dept.
Univ. of Southern California
ayanian@usc.edu

ABSTRACT

We present an algorithmic framework for automated construction using multiple identical robots. Our approach is based on the principle of tree-based dynamic programming and a concomitant idea of local search on trees to improve the quality of the generated plans. Inspired by the TERMES project of Harvard University, robots in this domain are required to gather construction blocks from a reservoir and coordinate with each other in building user-specified structures much larger than themselves. While the robots are roughly of the same size as the blocks, they can scale greater heights by using temporarily constructed ramps in the substructures. Our algorithm employs an inner loop in which the planning problem is solved by performing dynamic programming on a tree that spans the footprint of the user-specified structure. The outer loop of the algorithm furnishes a good tree for the inner loop. We show how we can search for a good tree in the outer loop using local search methods that yield significant improvements in plan quality. Synchronization rules are then applied to parallelize the execution of the generated plan for multiple identical robots.

General Terms

Algorithms, Performance, Experimentation

Keywords

Automated Construction, Multiple Identical Robots

1. INTRODUCTION

The Harvard TERMES project is intended to investigate how small reliable robots can cooperate in teams to build user-specified 3D structures much larger than themselves [2]. The hardware system consists of small autonomous mobile robots and a reservoir of passive "building blocks". The robots and blocks are endowed with mechanical features that make the system very reliable. Moreover, the robots are of roughly the same size as the blocks themselves; and yet, they can manipulate these blocks one at a time to build tall structures by stacking the blocks on each other and building ramps to scale greater heights. Multiple robots should be able to work with each other in a team to build a user-specified structure as cost-effectively and as fast as possible.

In [1], a polynomial-time tree-based algorithm is presented to solve planning problems in the TERMES domain. In particular, given a user-specified structure, the algorithm yields a cost-effective plan that attempts to minimize the number of pickup and drop-off operations on blocks. The inner loop of the algorithm solves the planning problem by conducting dynamic programming on a tree that spans the footprint of the user-specified structure. The outer loop of the algorithm furnishes a good tree for the inner loop. Put together, the algorithm exploits the *locality principle*: "distant substructures of a building are only loosely related to each other".

While the algorithm in [1] exploits the locality principle, it does not fully explore the space of all trees in the outer loop. Instead, it uses specific trees like the *minimum spanning tree* (MST) or its *reweighted* version (RMST). We show how we can search for a good tree in the outer loop using local search methods that yield significant improvements in plan quality. A second problem with the algorithm in [1] is that it generates a sequential plan meant to be executed by a single robot. In this paper, we also show how we can parallelize the generated plan using proper synchronization rules to make it viable for multiple robots. We henceforth use basic concepts and terminology introduced in [1].

2. LOCAL SEARCH ON TREES

Given a spanning tree T, the procedure 'Compute-Tree-Score' in Algorithm 1 calculates the cost of the construction plan generated by the inner loop if it conducted dynamic programming on T. Procedure 'Local-Search-on-Trees' in Algorithm 2 presents the local search procedure on the space of spanning trees. It takes as input a candidate spanning tree *seedTree* that acts as the starting point of the local search. Here, the graph G is a graphical representation of

*NSF 1409987, 1319966; MURI N00014-09-1-1031

Appears in: *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016), J. Thangarajah, K. Tuyls, C. Jonker, S. Marsella (eds.), May 9–13, 2016, Singapore.*

Algorithm 1: Procedure Compute-Tree-Score

Input: a spanning tree T of the workspace graph G
Output: the *score* of T

(1) Initialize *total* to 0.
(2) Generate markers on tree T.
(3) For each node N in T:
(a) Let $L_N = \{m_{(N,1)}, m_{(N,2)} \dots m_{(N,|L_N|)}\}$ be N's list of markers.
(b) For $i = 1, 2 \dots |L_N| - 1$:
(b1) $total = total + |m_{(N,i+1)} - m_{(N,i)}|$.
(4) Return *total*.

Algorithm 2: Procedure Local-Search-on-Trees

Input: a starting tree *seedTree* spanning G; the maximum number of flips *maxFlips*; the cycle length filter c
Output: an improved tree based on conducting local search on the space of spanning trees of G within the parameters *maxFlips* and c

(1) Set *currentTree* to *seedTree*.
(2) Set *minScore* to Compute-Tree-Score(*currentTree*).
(3) Set *flips* to 0.
(4) While *flips* < *maxFlips*:
(a) Let (u, v) be an edge in $G \backslash currentTree$ such that $dist_{currentTree}(u, v) < c$.
(b) Let graph H be $currentTree \cup \{(u, v)\}$.
(c) Let C be a cycle in H.
(d) Set lists *improvedList* and *equalsList* to empty lists.
(e) For each edge $e \neq (u, v)$ in C:
(e1) Let T be the tree $H \backslash \{e\}$.
(e2) If Compute-Tree-Score(T) < *minScore*, add e to list *improvedList*.
(e3) If Compute-Tree-Score(T) == *minScore*, add e to list *equalsList*.
(f) If *improvedList* is not empty, select an edge e' randomly from *improvedList*.
(g) Else if *equalsList* is not empty, select the least recently used edge e' from *equalsList* breaking ties randomly.
(h) Else with probability $0.5e^{-\frac{flips}{1000}}$, select a random edge $e' \neq (u, v)$ in C.
(i) If e' is still not defined from (f), (g) or (h), continue.
(j) $currentTree = currentTree \cup \{(u, v)\} \backslash \{e'\}$.
(k) Set *minScore* to Compute-Tree-Score(*currentTree*).
(l) Mark (u, v) and e' as being used in iteration number *flips*.
(m) Increment *flips*.
(5) Return *currentTree*.

the workspace matrix[1] with nodes corresponding to cells in the matrix and edges corresponding to adjacent cells. The procedure outputs an improved spanning tree that has a lower score than the starting tree.

Table 1 shows the significant benefits of our local search algorithm. Rows with 0 iterations indicate that no local search was done. The other rows indicate the score of the tree returned after 10000 (*maxFlips*) iterations starting from the specified *seedTree* with no cycle length filter. The '%Empty' column indicates the percentage of empty cells—cells where no towers stand—in the input matrix. We report only on the 30% empty and 70% empty cases due to space restrictions. We ran our experiments on input matrices of different sizes. Due to space restrictions, we report only on a 100 15×15 input matrices each with a maximum height of 15. The same trends are also observed in more elaborate data.

3. MULTIPLE IDENTICAL ROBOTS

The sequential plan generated by the algorithm in [1] is in the form of *waves*. Every odd numbered wave adds blocks

[1]input matrix padded with additional '0's on the boundary using a conservative estimate of how much space we need for constructing ramps [1]

%Empty	Iterations	*seedTree*	Field	Score
30	0	RBR	mean	7984.89
30	0	RBR	std	548.49
30	0	MST	mean	3900.07
30	0	MST	std	284.56
30	0	RMST	mean	3805.69
30	0	RMST	std	278.26
30	10000	RBR	mean	3412.73
30	10000	RBR	std	222.37
30	10000	MST	mean	3412.73
30	10000	MST	std	222.37
30	10000	RMST	mean	3412.73
30	10000	RMST	std	222.37
70	0	RBR	mean	4899.76
70	0	RBR	std	486.25
70	0	MST	mean	3173.90
70	0	MST	std	340.54
70	0	RMST	mean	2430.26
70	0	RMST	std	201.83
70	10000	RBR	mean	2119.50
70	10000	RBR	std	184.87
70	10000	MST	mean	2119.50
70	10000	MST	std	184.87
70	10000	RMST	mean	2119.50
70	10000	RMST	std	184.87

Table 1

Building Model	1 Agent	2 Agents	3 Agents
Eiffel Tower	9064	6758	6110
Empire State	5068	3740	3278
Giza Pyramid	5190	3344	2958
Taj Mahal	8064	5448	4756

Table 2

and every even numbered wave removes blocks. After the successful parallelization of a set of steps $s_{i+1}, s_{i+2} \dots s_{i+l_1}$ in the plan, the next set of steps $s_{i+l_1+1}, s_{i+l_1+2} \dots s_{i+l_1+l_2}$ for parallelization is heuristically chosen by examining the steps $s_{i+l_1+1}, s_{i+l_1+2} \dots$ one at a time and stopping when one of three rules for synchronization becomes effective. Here, $s_{i+l_1+l_2+1}$ is the first step after s_{i+l_1+1} such that either: (a) it belongs to a different wave compared to s_{i+l_1+1}, or (b) l_2 is equal to the maximum number of available robots, or (c) it is causally dependent on s_{i+l_1+j} for some $1 \leq j \leq l_2$. An action(step) a_2 with target node n_2 depends on an action a_1 with target node n_1 for its precondition in the plan only if a_1 occurs before a_2 and n_1 is an ancestor of n_2 in the tree along which the dynamic programming is carried out. Robots executing parallel actions traverse the spanning tree from the root to their target nodes (or vice versa) in parallel with the robot having to traverse the longest path starting first and all robots waiting until the last one is done.

Table 2 shows the number of parallel time steps needed for plan completion by different numbers of identical TERMES robots on models of world-famous buildings used in [1]. Experiments on random instances show similar patterns.

REFERENCES

[1] T. K. S. Kumar, S. Jung, and S. Koenig. A tree-based algorithm for construction robots. In *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling (ICAPS-2014)*, 2014.

[2] K. Petersen, R. Nagpal, and J. Werfel. Termes: An autonomous robotic system for three-dimensional collective construction. In *Proceedings of Robotics: Science and Systems (RSS-2011)*, 2011.