Intractability of Assembly Sequencing: Unit Disks in the Plane

Michael Goldwasser* and Rajeev Motwani**

Department of Computer Science
Stanford University
Stanford, CA 94305-9045
email: {wass,rajeev}@cs.stanford.edu

Abstract. We consider the problem of removing a given disk from a collection of unit disks in the plane. At each step, we allow a disk to be removed by a collision-free translation to infinity, and the goal is to access a given disk using as few steps as possible. This DISKS problem is a version of a common task in assembly sequencing, namely removing a given part from a fully assembled product. Recently there has been a focus on optimizing assembly sequences over various cost measures, however with very limited algorithmic success. We explain this lack of success, proving strong inapproximability results in this simple geometric setting. Namely, we show that approximating the number of steps required to within a factor of $2^{\log^{1-\gamma} n}$ for any $\gamma > 0$ is quasi-NP-hard. This provides the first inapproximability results for assembly sequencing, realized in a geometric setting.

As a stepping stone, we study the approximability of scheduling with AND/OR precedence constraints. The DISKS problem can be formulated as a scheduling problem where the order of removals is to be scheduled. Before scheduling a disk to be removed, a path must be cleared, and so we get precedence constraints on the tasks; however, the form of such constraints differs from traditional scheduling in that there is a choice of which path to clear. We prove our main result by first showing the similar inapproximability of this scheduling problem, and then by showing that a sufficiently hard subproblem can be realized geometrically using unit disks in the plane. Furthermore, our construction is fairly robust, in that is can be placed on a polynomially sized integer grid, it remains valid even when we consider only horizontal and vertical translations, and it also applies to axis-aligned unit squares and higher dimensions.

* Supported by ARO MURI Grant DAAH04-96-1-0007 and by NSF Award CCR-9357849, with matching funds from IBM, Mitsubishi, Schlumberger Foundation, Shell Foundation, and Xerox Corporation.

** Supported by an Alfred P. Sloan Research Fellowship, an IBM Faculty Partnership Award, an ARO MURI Grant DAAH04-96-1-0007, and NSF Young Investigator Award CCR-9357849, with matching funds from IBM, Mitsubishi, Schlumberger Foundation, Shell Foundation, and Xerox Corporation.

1 Introduction

The *assembly sequencing* problem consists of taking a geometric description of a product, consisting of a set of parts, and producing a sequence of collision-free operations which results in the (dis)assembly of the product. Many complexity measures for judging the quality of a sequence were discussed in a precursor to this work [17], where we prove strong inapproximability for a non-geometric generalization of the sequencing problem. In this paper, we show that strong lower bounds can be realized geometrically, for the problem of planning for the removal of a given part from a fully assembled product. This problem is motivated by the issues of maintenance and of recycling. The classic maintenance example is the need to replace a spark plug without taking the entire car apart. A classic recycling example is to strip down an old computer for a valuable part with minimal effort. Unfortunately, there has been little success algorithmically for optimizing (dis)assembly sequences over complexity measures, and several current assembly sequencing packages must rely on either a brute force search through all possibilities, or on heuristic searches with no performance guarantees. Our work proves the difficulty of finding optimal or near-optimal cost assembly sequences, under much simpler geometric conditions than are needed by industrial assembly sequences.

Specifically, we examine what we term the DISKS problem. The goal is to remove a key disk from a collection of unit disks in the plane, where each step allows for a collision-free translation of a single disk to infinity. The cost of a solution is equal to the number of steps required to remove the key disk. We prove that it is quasi-NP-hard to approximate this DISKS problem to within a factor of $2^{\log^{1-\gamma} n}$ for any $\gamma > 0$. These hardness results are the first such inapproximability results involving the cost of assembly sequences, to be realized geometrically.

To prove our results, we study the problem of scheduling with AND/OR precedence constraints, a generalization of the DISKS problem. We prove the inapproximability of this AND/OR scheduling problem, previously shown only to be NP-hard, using a reduction from the LABELCOVER$_{\min}$ problem [3,4]. Moreover, we show that the hardness of AND/OR scheduling applies to a more restricted version which can then be realized geometrically as our DISKS problem. Our construction can be placed on a polynomially sized grid, and our results are valid even when translations are limited to two directions. (Limited to one direction, the problem is trivially solvable.) These results also generalize to axis-aligned unit squares as well as to higher dimensions. Full details of this paper are included in a more extensive study of the approximability of cost measures for assembly sequencing [16].

2 Previous Work

Assembly Sequencing: The use of automation in assembly sequencing has increased rapidly over the years [6,11,24–26,35,43,44,46]. Theoretical results show that assembly sequencing, in its most general form, is intractable [27,31,32,37,46],

and thus many researchers began considering restricted, yet still interesting, versions of the problem. For many of these restricted settings, polynomial algorithms have been designed which find an assembly sequence if one exists [1,19,22,43,45]. There are also algorithms which enumerate all possible assembly sequences [11], however there may be exponentially many such sequences for a product.

A logical continuation to this success is to use automated reasoning to find the "best" assembly sequence under certain complexity measures. In fact, the IEEE Technical Committee on Assembly and Task Planning summarized the current state of assembly sequencing by explaining [18], "... after years of work in this field, a basic planning methodology has emerged that is capable of producing a feasible plan ... The challenges still facing the field are to develop efficient and robust analysis tools and to develop planners capable of finding optimal or near–optimal sequences rather than just feasible sequences." Unfortunately, meeting this challenge has been difficult. Several complexity measures for assembly sequencing have been suggested, motivated strongly by industrial applications [7,17,45,46]. In fact, some current software systems offer the user the option of optimizing the sequence over a choice of complexity measures [30,40], however these systems currently must rely either on heuristics with no performance guarantees on the cost of the sequence, or else on possibly exponential search techniques to find the true optimal. The NP-completeness of exactly optimizing several measures is shown in a symbolic framework [46], and the inapproximability of several measures, including the one considered here, are shown in a graph-theoretic generalization of assembly sequencing [17]. The strongest of these lower bounds, however, are not realized geometrically. For a restricted class of inputs which have a so-called "total ordering" property, a greedy algorithm is given which claims to produce the minimal length sequence to remove any given part [47], however the required property does not have a clean definition, and as our results will show, this problem is quite difficult in general.

Approximability Theory. As our optimization problems turn out to be NP-hard, we approach the problems using techniques common to the theory of approximability [4,13,28,36]. Since we cannot expect to find the optimal sequence in polynomial time, we look for a polynomial time *approximation algorithm* which returns a solution whose cost can be bounded by some function of the true optimal cost. A standard measure for the quality of an approximation algorithm is the *approximation ratio* between the cost of the solution returned by the algorithm versus the cost of the optimal solution. Although all NP-complete decision problems can be reduced to one another, the approximability of such problems can be quite different, ranging from problems which can be approximated arbitrarily close to optimal, to problems where getting even a very rough approximation is already NP-hard. Many researchers have worked towards classifying the approximability of different NP-hard problems. We consider four broad classes defined in [4], which group problems based on the strength of the inapproximability results which have been proven. Class I includes all problems for which approximating the optimal solution to within a factor of $(1+\epsilon)$ is NP-hard for some $\epsilon > 0$ (e.g., MAX-3SAT [38]). Class II groups those problems for which it

is quasi-NP-hard[1] to achieve an approximation ratio of $c \cdot \log n$ for some $c > 0$ (e.g., SET COVER [12]). For problems in Class III, it is quasi-NP-hard to achieve a $2^{\log^{1-\gamma} n}$ factor[2] approximation for any $\gamma > 0$ (e.g., LABELCOVER [3]). Finally, Class IV consists of the hardest problems, namely those for which it is NP-hard to achieve an n^{ϵ} approximation factor for some $\epsilon > 0$ (e.g., CLIQUE [23]). Our work places both the AND/OR scheduling problem and the DISKS problem into Class III of this hierarchy.

Computational Geometry. Assembly sequencing is an intriguing combination of a combinatorial and geometric problem. Quite naturally, research from computational geometry relates very closely to assembly sequencing. The separability of objects has been well studied in the geometric community [8–10,20,41,42]. In a single direction, a *depth order* of a set of parts is an ordering of the parts which allows for collision-free translations of the individual parts to infinity. A classic result states that given a collection of convex shapes in two dimensions, for any translational direction there exists some ordering, such that the parts can be translated away one at a time [20]. Similar separability issues are studied in two dimensions for more general classes of shapes, such as monotone or star-shaped polygons [42]. For a collection of balls in $\mathcal{R}^d$, there exist at least $d+1$ balls, each of which can be translated to infinity in some direction [8]. Unfortunately, there exist collections of unit disks in the plane for which the removal of a certain disk may require the prior removal of $\Omega(n)$ other disks [21]. Also, there exists a set of convex parts in three dimensions which cannot be disassembled using two hands, with only translations, or even generalized rotations and translations [41].

A surprising element of our problem is that the general lower bounds can be realized for a simple geometric setting consisting of a collection of unit disks in the plane. More often than not, an optimization problem becomes significantly easier when its input is restricted to a geometric setting. For example, there exists some $c > 0$ for which achieving a $(1 + c)$-approximation for the METRIC TRAVELING SALESMAN problem is NP-hard [39], however in the Euclidean plane, TSP can be approximated to within $(1 + \epsilon)$ for all $\epsilon > 0$ [2]. Similarly, achieving an n^{ϵ}-approximation for MINIMUM INDEPENDENT SET is NP-hard [23], however for planar graphs, MIS can be approximated to within $(1 + \epsilon)$ [5]. Similar results hold for most optimization problem when restricted to planar graphs [33]. There exists a $\ln n$ lower bound for approximating the SET COVER problem [12], however the RECTANGLE COVER problem, covering a set of axis-aligned rectangles with minimum number of points, has no such inapproximability results [36].

---

[1] That is, this would imply NP $\subseteq$ DTIME($n^{\mathrm{poly}(\log n)}$). "A proof of quasi-NP-hardness is good evidence that the problem has no polynomial-time algorithm." [4]

[2] This factor, $2^{\log^{1-\gamma} n}$, lies between polynomial and polylogarithmic in that $2^{\log^{1-\gamma} n} = o(n^{\epsilon})$ for any $\epsilon > 0$, and $2^{\log^{1-\gamma} n} = \omega(\log^{c} n)$ for any constant c.

3 Scheduling with AND/OR Precedence Constraints

We can view an instance of the DISKS problem quite naturally as a scheduling problem. The removal of each part is thought of as a task which can be scheduled, and the goal is to schedule a key task as early as possible. The cost of a solution is equal to the total number of tasks scheduled.

Of course, our tasks have certain precedence constraints relating their order of removal. That is, it may be the case that a certain part cannot be removed until after some other parts are removed. What distinguishes this setting from more traditional scheduling is the form of the precedence constraints. Commonly, a task may have what we term an AND-precedence constraint, in that it has an associated set of tasks, all of which must be scheduled before that task [13]. Unfortunately, this is not the case in our assembly sequencing problem. When considering a single direction, if a part is to be removed, then indeed there is a clear set of associated parts which block the removal, and thus all of these parts must be removed prior to removing our part in that direction. However, we may choose to remove that same part in some other direction, in which case a different set of parts may block the removal.

Instead, our problem can be viewed as an instance of scheduling with AND/OR precedence constraints, where the OR's allow us to offer a choice of directions. We will consider scheduling where every task has a set of direct predecessors, and each task is either an AND-task, in which case it cannot be scheduled until after all of its predecessors, or the task is an OR-task, in which case it cannot be scheduled until after at least one of its predecessors. For our setting, we consider a single processor and unit processing time for all tasks. It is worth nothing that with classical AND precedence constraints, this problem of minimizing the number of scheduled tasks can be solved exactly, in polynomially time by computing a depth order.

Notation and Definitions. We define the problem of scheduling with AND/OR precedence constraints as follows. The input contains a set of tasks, $\mathcal{T}$. Each task, $t_i \in \mathcal{T}$, is labeled as either an AND-*task* or an OR-*task*. Each task, $t_i \in \mathcal{T}$, has an associated set of tasks, P_i, as direct *predecessors*; we refer to $|P_i|$ as the *degree* of the task. An AND-task, t_i, cannot be scheduled until after *all* tasks in P_i. An OR-task, t_j, cannot be scheduled until after *at least one* task of P_j. The *max* AND-*degree* of an instance is the maximum size $|P_i|$ over all AND-tasks t_i. The *max* OR-*degree* of an instance is the maximum size $|P_j|$ over all OR-tasks t_j.

The constraints can be represented by a *precedence graph*, with a node for each task, t_i, and a directed edge from t_i to t_j whenever $t_j \in P_i$, is a direct predecessor of t_i. A *leaf-task* is one with $P_i = \emptyset$, and thus no outgoing edges. Such a task can be scheduled at any time. We say that an instance of AND/OR scheduling has *partial-order* precedence constraints if there are no cycles in the precedence graph. We say an instance of AND/OR scheduling has *internal-tree* precedence constraints if there are no cycles, and if all non-leaf nodes have at most one incoming edge.

The goal for this problem is to successfully schedule a specific task, and the cost is equal to the total number of tasks which must be scheduled. We consider a single processor and unit processing time for all tasks. Additionally, we will consider the problem of minimizing the number of scheduled *leaves*, when constrained to internal-tree precedence constraints. Notice that internal-tree precedence constraints define a monotone, boolean formula on the leaf nodes, in which setting a leaf's variable to "one" signifies that the leaf will be scheduled. Minimizing the number of scheduled leaves is equivalent to satisfying a monotone, boolean formula with the minimum number of ones. We are unaware of any previous results for this exact approximation problem. Minimizing the number of ones in satisfying a 3CNF formula is known to be $n^{0.5-\epsilon}$-hard to approximate [29], and related minimization problems are studied in [34].

Previous Work. A model for scheduling with AND/OR precedence constraints was introduced in [14,15], however with one key difference. In this previous work, only the case of partial order precedence constraints is considered. Notice that in traditional scheduling, with AND-precedence constraints, the existence of a cycle in the precedence constraints makes the scheduling problem infeasible, and thus a partial order is a standard assumption. With AND/OR constraints, this absence of cycles is no longer a necessary condition for the existence of a valid solution. In fact, cycles will often exist in instances drawn from assembly sequencing, as it may be the case that part A blocks part B in one direction, part B blocks part C in another, and part C blocks part A in a third direction. For this reason, we make no apriori assumptions about the structure of the precedence constraints.

The work of [14,15] studies a larger variety of settings, including multiple processors, deadlines, and individual processing times. They prove the NP-hardness of finding feasible schedules in many setting which were polynomially solvable with more traditional AND-precedence constraints, however they do not consider the approximability of the corresponding optimization problems.

3.1 Inapproximability of AND/OR Scheduling

Theorem 1. *It is quasi-NP-hard to find a solution to any of the following problems which is within a factor of* $2^{\log^{1-\gamma} n}$ *of the optimal solution, for any* $\gamma > 0$.

- *Minimizing the number of* leaves *scheduled, for* AND/OR *scheduling with internal-tree precedence constraints.*
- *Minimizing the number of* leaves *scheduled, for* AND/OR *scheduling with internal-tree precedence constraints, and max-degree bounded by two.*
- *Minimizing the number of* tasks *scheduled, for general* AND/OR *scheduling.*
- *Minimizing the number of* tasks *scheduled, for general* AND/OR *scheduling with max-degree bounded by two.*

Proof. We begin by considering the AND/OR scheduling problem when restricted to *internal-tree* precedence constraints, when charged for the number of *leaves* scheduled. We show the inapproximability of this problem by showing that the

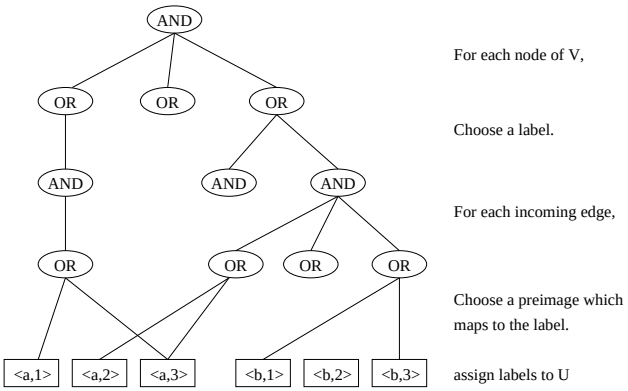

Fig. 1. LABELCOVER as AND/OR scheduling with internal-tree precedence

LABELCOVER$_{\min}$ problem is a special case. The LABELCOVER$_{\min}$ problem, defined in [4], is an artificial generalization of the SET COVER problem introduced in approximability theory. The input is a regular bipartite graph, $G = (U, V, E)$, a set of labels $\{1, 2, \ldots, N\}$, and a partial function $\Pi_e : \{1, 2, \ldots, N\} \longrightarrow \{1, 2, \ldots, N\}$ for each edge $e \in E$. A labeling associates a non-empty set of labels with every vertex in $U \cup V$. It is said to *cover* an edge $e = (u, v)$, if for every label b assigned to v, there is some label a assigned to u such that $\Pi_e(a) = b$. The goal of LABELCOVER$_{\min}$ is to give a labeling which covers all edges, while minimizing the total number of labels assigned to nodes of U.

Given an instance of LABELCOVER$_{\min}$, we express it as an instance of AND/OR scheduling with internal-tree precedence constraints. The AND/OR instance has five levels, which alternate between AND-nodes and OR-nodes. The highest level contains solely the root of the internal-tree, and the lowest level contains exactly the leaves. The tasks at the five levels are as follows:

- The first level has a single AND-node, which is the root of the internal-tree. This task enforces that for a valid labeling, every node in V must have a non-empty set of labels.
- The second level has an OR-node for each vertex in V. This nodes requires that for a given node v to have a non-empty label set, at least one label must be assigned to it.
- The third level has an AND-node for each pair $\langle v, l' \rangle$, where $v \in V$, and $l' \in \{1, \ldots, N\}$. This node signifies that for label l to be assigned to vertex v, it must be the case that for each edge $e = (u, v)$ incident to v, the mapping Π_e on that edge, must respect the labeling.
- The fourth level has an OR-node for each pair $\langle e, l' \rangle$, where $e = (u, v)$ is an edge, and l' is a label. If l' is to be assigned to v, then edge e cannot be covered unless one of the pre-images of l' from mapping Π_e is assigned to u.
- The fifth level has a leaf for each pair $\langle u, l \rangle$, and corresponds to label l being assigned to vertex u.

This completes the construction. It can be seen that there is a one-to-one correspondence between valid labeling in the LABELCOVER$_{\min}$ instance and valid solutions to the AND/OR scheduling instance. It is easy to verify that the AND/OR instance has internal-tree precedence constraints. Notice that the number of non-leaf tasks in this construction is polynomially bounded in the size of the LABELCOVER$_{\min}$ instance (namely, in $|U|$, $|V|$ and N). Combining this with the result of [4] which proves a similar lower bound for the approximability of LABELCOVER$_{\min}$, we get that it is quasi-NP-hard to achieve an $2^{\log^{1-\gamma} n}$ approximation for any $\gamma 0$, when minimizing the number of leaves with internal-tree precedence constraints.

Given an instance with unbounded degree, we can bound the in-degree in the obvious way, by replacing each internal node with a tree of bounded degree nodes. Assume there were originally I internal nodes and L leaves, and that I is polynomially bounded in L. The maximum fan-in for any node is at most $(I+L)$, and thus that node must be replaced by a tree of at most $(I+L)$ nodes, each with fan-in two. The new instance has $I(I+L)$ internal nodes, which is still polynomially-bounded in L.

The only difficulty in switching from the measure of counting scheduled leaves to counting all scheduled tasks is that the overhead of the internal nodes may have a significant cost, changing our approximation ratio. This can be remedied quite easily. Assume we have a hard instance of AND/OR scheduling with internal-tree precedence constraints, containing I internal nodes and L leaves. We convert this to a general instance of AND/OR scheduling by hanging from each leaf a chain of I new nodes. Notice that the OR-degree is not increased, although this new instance no longer has internal-tree precedence constraints. In this way, the problematic additive cost can be made arbitrarily small, and so the only issue in completing the proof is to address the input size. Originally, the value n was assumed to be the number of leaves. In our new instance, the total number of nodes is n', however we can rely on the fact that $n' = \text{poly}(n)$, and thus an approximation ratio of $2^{\log^{1-\gamma} n'}$ is less than a ratio of $2^{\log^{1-\gamma'} n}$ for some $\gamma' > 0$.

4 The DISKS Problem

Theorem 2. *It is quasi-NP-hard to approximate the* DISKS *problem to within a factor of* $2^{\log^{1-\gamma} n}$ *for any* $\gamma > 0$. *This bound also applies if we consider only translations along the positive* X*-axis and* Y*-axis. Additionally, this construction generalizes to axis-aligned unit squares, and to higher dimensions.*

Proof. Assume we are given a hard instance from Theorem 1, of AND/OR scheduling with internal-tree precedence constraints, and OR-degree bounded by two (we do not require such a bound on the AND-degree). Without loss of generality, we assume that OR-nodes rely only on internal nodes.

We construct the following instance of the DISKS problem. Our scene consists entirely of disks with radius one, whose centers lie on a polynomially-sized, integer grid. We prove this result directly for the case where only two directions

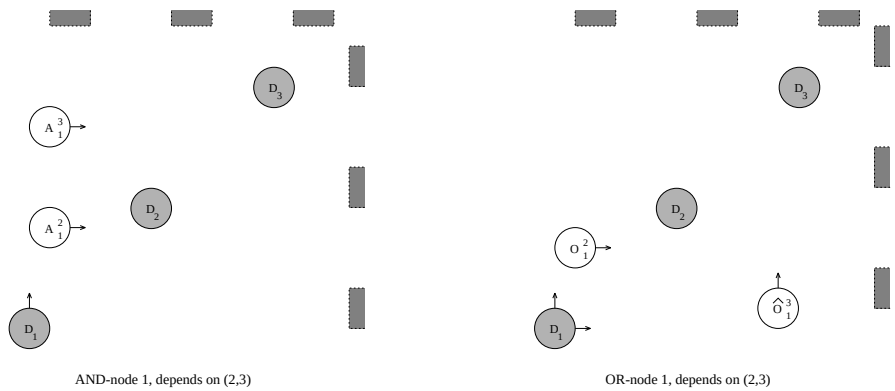

Fig. 2. Internal node mechanisms

of translations are allowed, namely North and East. We place a wall of width $2W$ around the perimeter of our working area which we consider immovable. We will place some holes in the wall, described later, which allow a clear path out for some disks. We consider our main working area to have two sections, one for the mechanisms involving the interior nodes, and the second section for the leaf node mechanisms.

First we describe the mechanism involving the internal nodes. Since the internal-tree defines a partial order on these nodes, we can number the internal nodes, $T_1, \ldots, T_I$ so that if an internal node depends on another internal node, it will have a higher index. For each internal node, T_i, we create a disk, D_i, centered at $(6i, 6i)$. We define the wall to the North by placing a column of W disks with x-coordinates centered at $6i+2$ for each disk D_i, assuring us that the disk itself has an "escape route" to the North. For the East wall, we place a row of disks centered at y-coordinate $6i+2$ in the case that disk D_i is an OR-disk, or else at $6i+1$ in the case that disk D_i is an AND-disk. In this way, we assure an additional escape passage to the East for an OR-disk, but not for an AND-disk.

Next, we add in additional disks to enforce the precedence constraints. For AND-node, T_i, blocked by node $T_k \in P_i$ (and thus $i < k$), we add a disk A_i^k centered at $(6i+1, 6k-1)$, which will be forced to the East by our previous placement of the walls. For an OR-node, T_i, which depends on 2 nodes, T_k and T_l, we create two new disks, O_i^k located at $(6i+1, 6k-1)$, which will be forced East by our walls, and $\hat{O}_i^l$ located at $(6l-1, 6i+1)$, which will be forced North. The entire internal node mechanisms are contained in a $(6I+1) \times (6I+1)$ square. Examples are given in Figure 2.

The section for the leaf mechanisms begins at height $6(I+1)$ so as to be higher than the internal mechanisms. We can number the leaf nodes in any order, and we create a separate mechanism for each leaf in a strip of height $2I$. For a given leaf, L_a, we create what we term a *blockade*, to the right of this strip. The blockade consists first of a diagonal chain of to the Northeast of height $2I$, followed by a horizontal chain of B disks to the East of the end of the first chain

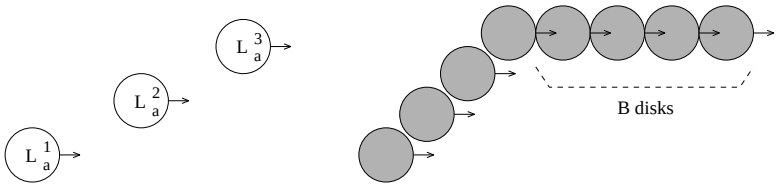

Fig. 3. Leaf node mechanism

(where B is determined later). The disk beginning the blockade is centered at $(6(I+1), 6(I+1)+Ia)$. The wall to the East of the blockade is removed, allowing the disks of the blockade an escape. For any disk located in the horizontal strip associated with L_a, escaping to the East will require an additional cost of at least B to break through the blockade. However this cost is only charged once per blockade, after which any disks in the horizontal strip may escape. Now, for every internal node T_i which depends on leaf L_a, we create a disk L_a^i, located at $(6i+1, 6(I+1)+Ia+2i)$, which is forced East by the walls. Figure 3 shows an example of a leaf mechanism.

To complete the construction, we set the blockade value, $B = 4I(L+I)$, to be greater than the total number of disks in the remainder of the internal and leaf mechanisms combined. In this way, the number of blockades removed dominates any additive costs in the rest of the construction. Finally, we assign $W = B(L+1)$, so that the cost of removing all non-wall disks is less than the cost of digging a single new hole through any part of the wall. For this reason, we may assume without loss of generality that any solution to this DISKS instance has cost at most W. Finally, we note that the wall has perimeter which is $O(BL)$, and hence the total number of disks in our construction is polynomially bounded. An example of the final construction is given in Figure 4.

It is not hard to verify that for this DISKS instance, a solution for removing the root disk with cost at most kB can be translated to an AND/OR solution of cost at most k. Similarly, an AND/OR solution of cost k can be translated to a DISKS solution with cost less than $(k+1)B$. Therefore, approximating the DISKS problem to within a factor of $2^{\log^{1-\gamma} n}$ for any $\gamma > 0$ is quasi-NP-hard, as the additive error and the polynomial increase of the input size disappear by adjusting γ.

Our proof shows the hardness of the DISKS problem when translations are limited to the North and East. In fact, if we allow translations in arbitrary directions, the theorem holds using this same construction. Furthermore, even if we are not restricted to linear moves, we could prove the same lower bound for minimizing the number of disks removed.

It is also easy to see that the disks can be replaced by axis-aligned, 2×2 squares and the construction still holds. For higher dimensions, the walls can be extended to block any useful motions in other dimensions, while still using polynomially many disks.

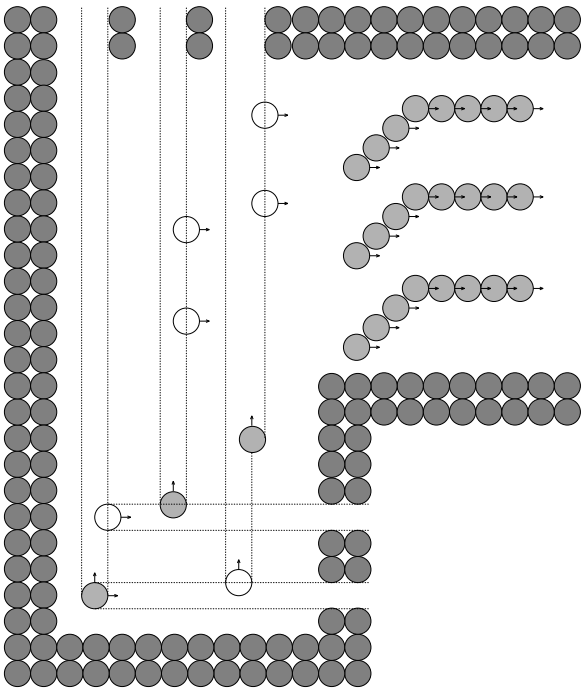

Fig. 4. A complete DISKS construction

5 Future Directions

Open directions for future research fall into three quite different directions. First, we offer no non-trivial (i.e. $o(n)$) approximation algorithms for either the DISKS problem or any variants of AND/OR scheduling. It so happens that there is no known, non-trivial approximation algorithm for LABELCOVER, the "easiest" of these problems. However our reductions are a bit misleading in this respect, because the polynomial blowup in the input size. It may be possible to achieve a non-trivial upper bound for many of our variants of AND/OR scheduling, without providing such an upper bound on LABELCOVERor the other variants. For example, the existence of an $n^{1/2}$-approximation for the DISKS problem would guarantee through our reductions, an n^{β}-approximation for $\text{LABELCOVER}_{\min}$, however it may be that $\beta > 1$, in which case the approximation would be trivial.

Secondly, it is open to strengthen the lower bounds for any of these problems. The $\text{LABELCOVER}_{\min}$ results provide our strongest results even for the most general AND/OR scheduling problem, yet there is reason to believe this may be an even more difficult problem. It is already conjectured that LABELCOVER is truly n^{ϵ}-hard to approximate for some $\epsilon > 0$ [4], a result which would carry over through all of our reductions. However, we feel that the study of AND/OR scheduling may provide more meaningful implications in the approximation com-

plexity hierarchy. It may be possible to strengthen the lower bounds for AND/OR scheduling without necessarily settling the LABELCOVER conjecture.

We examined a very structured class of instances of AND/OR scheduling which had what we termed *internal-tree* precedence constraints, and we considered charging only for the *leaves* that are scheduled. Without a bound on the out-degree, we can collapse internal nodes so that our tree has alternating levels of AND-nodes followed by OR-nodes, with a final level of leaves. If we restrict the number of alternating levels to one, this problem becomes *exactly* SET COVER, and thus lies in Class II of the approximation hierarchy. When the number of alternating levels is equal to two, we saw in Figure 1, that this problem already captures LABELCOVER, and thus lies in Class III. However it is not at all clear that this problem is equivalent to LABELCOVER, as it may capture more instances. Furthermore, what happens when we go to three full alternations, or to an arbitrary depth internal tree? Does this hierarchy collapse at some point, and if so when? Can the inapproximability bounds be strengthened for any of these versions? What about when no constraints at all are placed on the structure of the precedence graph?

Finally, the DISKS problem is just one version of a collection of optimization problems for assembly sequencing. A more complete framework for analyzing the approximability of many other cost measures for assembly sequencing problems is presented in [16]. In that work, many strong inapproximability results are shown in a non-geometric generalization, however with the exception of the DISKS setting, the geometric lower bounds shown for other cost measures are far weaker than their non-geometric counterparts. A complete list of related problems and open questions is given there.

Acknowledgments

The authors wish to thank Danny Halperin for suggesting that we consider the scenario with unit disks. Also, thanks go to Chandra Chekuri and Sanjeev Khanna for their involvement in the study of the AND/OR scheduling model. Finally, to Cyprien Godard, Jean-Claude Latombe, G. Ramkumar, Bruce Romney, and Randy Wilson, for their involvement in earlier parts of this work as well as for their many helpful discussions and suggestions.

References

1. P. Agarwal, M. de Berg, D. Halperin, and M. Sharir. Efficient generation of k-directional assembly sequences. In *Proc. 7th ACM Symp. on Discrete Algorithms*, pages 122–131, 1996.
2. S. Arora. Polynomial-time approximation schemes for Euclidean TSP and other geometric problems. In *Proc. 37th Symp. on Found. Comput. Sci.*, pages 1–11, 1996.
3. S. Arora, L. Babai, J. Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes and linear equations. In *Proc. 34th Symp. on Found. Comput. Sci.*, pages 724–733, 1993.

4. S. Arora and C. Lund. Hardness of approximations. In D. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*. PWS Publishing Company, Boston, MA, 1996.
5. B. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. ACM*, 41(1):153–180, 1994.
6. D. Baldwin. Algorithmic methods and software tools for the generation of mechanical assembly sequences. M.Sc. thesis, MIT, Cambridge, MA, 1990.
7. G. Boothroyd. *Assembly Automation and Product Design*. Marcel Dekker, Inc., New York, NY, 1991.
8. R. Dawson. On removing a ball without disturbing the others. *Mathematics Magazine*, 57(1):27–30, 1984.
9. M. de Berg, M. Overmars, and O. Schwarzkopf. Computing and verifying depth orders. *SIAM J. Comput.*, 23(2):432–446, 1994.
10. F. Dehne and J.-R. Sack. Translation separability of polygons. *Visual Computer*, 3(4):227–235, 1987.
11. T. D. Fazio and D. Whitney. Simplified generation of all mechanical assembly sequences. *IEEE Trans. on Robotics and Automation*, 3(6):640–658, 1987.
12. U. Feige. A threshold of $\ln n$ for approximating set cover. In *Proc. 28th ACM Symp. Theory Comput.*, pages 314–318, 1996.
13. M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, NY, 1979.
14. D. Gillies. *Algorithms to schedule tasks with AND/OR precedence constraints*. Ph.D. thesis, University of Illinois, Urbana, IL, 1993.
15. D. Gillies and J. Liu. Scheduling tasks with AND/OR precedence constraints. *SIAM J. Comput.*, 24(4):797–810, 1995.
16. M. Goldwasser. *Complexity Measures for Assembly Sequencing*. Ph.D. thesis, Dept. Comput. Sci., Stanford Univ., Stanford, CA, 1997.
17. M. Goldwasser, J.-C. Latombe, and R. Motwani. Complexity measures for assembly sequences. In *Proc IEEE Int. Conf. on Robotics and Automation*, pages 1581–1587, 1996.
18. S. Gottschlich, C. Ramos, and D. Lyons. Assembly and task planning: A taxonomy. *IEEE Robotics and Automation Magazine*, 1(3):4–12, 1994.
19. L. Guibas, D. Halperin, H. Hirukawa, and J.-C. L. R. Wilson. A simple and efficient procedure for polyhedral assembly partitioning under infinitesimal motions. In *Proc IEEE Int. Conf. on Robotics and Automation*, pages 2553–2560, 1995.
20. L. Guibas and F. Yao. On translating a set of rectangles. In F. Preparata, editor, *Computational Geometry*, Advances in Computing Research, pages 61–77. JAI Press Inc., 1983.
21. D. Halperin. Personal communication. 1995.
22. D. Halperin and R. Wilson. Assembly partitioning along simple paths: the case of multiple translations. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 1585–1592, 1995.
23. J. Hstad. Clique is hard to approximate within $n^{1-\epsilon}$. In *Proc. 37th Symp. on Found. Comput. Sci.*, pages 627–636, 1996.
24. R. Hoffman. A common sense approach to assembly sequence planning. In *Computer-Aided Mechanical Assembly Planning*, pages 289–314. Kluwer Academic Publishers, Boston, 1991.
25. L. Homem de Mello and A. Sanderson. *Computer-Aided Mechanical Assembly Planning*. Kluwer Academic Publishers, Boston, 1991.

26. L. Homem de Mello and A. Sanderson. A correct and complete algorithms for the generation of mechanical assembly sequences. *IEEE Trans. on Robotics and Automation*, 7(2):228–240, 1991.
27. J. Hopcroft, J. Schwartz, and M. Sharir. On the complexity of motion planning for multiple independent objects: P-space hardness of the "Warehouseman's Problem". *Int. J. Robotics Research*, 3(4):76–88, 1984.
28. D. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. Systems Sci.*, 9:256–278, 1974.
29. V. Kann. Polynomially bounded minimization problems which are hard to approximate. In *Automata, Languages and Programming (Proc. 20th ICALP)*, volume 700 of *Lecture Notes in Computer Science*, pages 52–63. Springer-Verlag, 1993.
30. S. Kaufman, R. Wilson, R. Jones, T. Calton, and A. Ames. The Archimedes 2 mechanical assembly planning system. In *Proc IEEE Int. Conf. on Robotics and Automation*, pages 3361–3368, 1996.
31. L. Kavraki and M. Kolountzakis. Partitioning a planar assembly into two connected parts is NP-complete. *Information Processing Letters*, 55(3):159–165, 1995.
32. L. Kavraki, J.-C. Latombe, and R. Wilson. Complexity of partitioning an assembly. In *Proc. 5th Canad. Conf. Comput. Geom.*, pages 12–17, Waterloo, Canada, 1993.
33. S. Khanna and R. Motwani. Towards a syntactic characterization of PTAS. In *Proc. 28th ACM Symp. Theory Comput.*, pages 329–337, 1996.
34. S. Khanna, M. Sudan, and L. Trevisan. Constraint satisfaction: The approximability of minimization problems. In *Proc. 12th IEEE Comput. Complexity Conference*, 1997.
35. S. Lee and Y. Shin. Assembly planning based on geometric reasoning. *Computers and Graphics*, 14(2):237–250, 1990.
36. R. Motwani. Approximation algorithms. Stanford Technical Report STAN-CS-92-1435, 1992.
37. B. Natarajan. On planning assemblies. In *Proc. 4th ACM Symp. on Computational Geometry*, pages 299–308, 1988.
38. C. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *J. Comput. Systems Sci.*, 43(3):425–440, 1991.
39. C. Papadimitriou and M. Yannakakis. The traveling salesman problem with distances one and two. *Mathematics of Operations Research*, 18(1):1–11, 1993.
40. B. Romney, C. Godard, M. Goldwasser, and G. Ramkumar. An efficient system for geometric assembly sequence generation and evaluation. In *Proc. ASME Int. Computers in Engineering Conference*, pages 699–712, 1995.
41. J. Snoeyink and J. Stolfi. Objects that cannot be taken apart with two hands. In *Proc. ACM Symp. on Computational Geometry*, pages 247–256, 1993.
42. G. Toussaint. Movable separability of sets. In G. Toussaint, editor, *Computational Geometry*, pages 335–375. North-Holland, Amsterdam, Netherlands, 1985.
43. R. Wilson. *On Geometric Assembly Planning*. Ph.D. thesis, Dept. Comput. Sci., Stanford Univ., Stanford, CA, 1992. Stanford Technical Report STAN-CS-92-1416.
44. R. Wilson, L. Kavraki, J.-C. Latombe, and T. Lozano-Pérez. Two-handed assembly sequencing. *Int. J. of Robotics Research*, 14(4):335–350, 1995.
45. R. Wilson and J.-C. Latombe. Geometric reasoning about mechanical assembly. *Artificial Intelligence*, 71(2):371–396, 1994.
46. J. Wolter. *On the Automatic Generation of Plans for Mechanical Assembly*. Ph.D. thesis, University of Michigan, 1988.
47. T. Woo and D. Dutta. Automatic disassembly and total ordering in three dimensions. *J. Engineering for Industry*, 113(2):207–213, 1991.