Intractability of Assembly Sequencing: Unit Disks in the Plane

Michael Goldwasser *   Rajeev Motwani †

Department of Computer Science
Stanford University
Stanford, CA 94305-9045

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*. These inapproximability results, to the best of our knowledge, are the strongest hardness results known for any purely combinatorial problem 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 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. This is one of the important problems arising in manufacturing processes where computational geometry is expected to have a profound impact [11]. Many complexity measures for judging the quality of a sequence are discussed in [17], where strong inapproximability results ae proven 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 the problem of removing a key disk from a collection of unit disks in the plane, where each step allows for a collision-free translation of a disk to infinity. We consider the optimization problem of minimizing the total number of disks which are removed in the process. Unfortunately, we prove that approximating this DISKS problem to within a $2^{\log^{1-\gamma} n}$-factor[1] is quasi-NP-hard[2] for any $\gamma > 0$. These hardness results are the first such inapproximability results involving the cost of assembly sequences, to be realized geometrically. Furthermore, to the best of our knowledge, this provides the strongest lower bounds for the approximability of any purely combinatorial problem in a geometric setting.

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 known 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. This reduction is 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.

Open directions for future research fall into two different areas. First, devising a non-trivial upper bound for the DISKS problem (and other assembly sequencing problems) remains an important goal. Although the results of this paper prove a discouraging lower bound, assembly sequencing is a preprocessing phase for a long and expensive manufacturing process, and so even a coarse approximation (e.g., an $O(\sqrt{n})$-approximation) would be of great value when it comes time to perform the sequence in mass quantities. A second direction for research involves proving n^ϵ lower bounds on the approximability of any of these problems. Additionally, we feel that the study of AND/OR scheduling may provide more meaningful implications in the complexity hierarchy.

---

[1] 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.

[2] 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 Previous Work

Assembly Sequencing. The use of automation in assembly sequencing has increased rapidly over the years [6, 12, 24, 25, 26, 33, 34, 43, 44, 46, 47]. Theoretical results showed that assembly sequencing, in its most general form, is NP-complete [27, 30, 36, 47], 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 [12], 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, 47]. In fact, some current software systems offer the user the option of optimizing the sequence over a choice of complexity measures [29, 40], however these systems must rely on either a brute force search of the entire space, or else A^*-type searches without performance guarantees. The NP-completeness of exactly optimizing several measures is shown in a symbolic framework [47]. Furthermore, the approximability of several measures, including the number of removed parts, are examined in a precursor of this work [17], as lower bounds for the approximability of several measures are shown in a graph-theoretic generalization of assembly sequencing. 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 [48], however the required property does not have a clean definition, and as our results will show, this problem is quite difficult.

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, 14, 28, 35]. 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 [37]). Class II groups those problems for which it is quasi-NP-hard to achieve an approximation ratio of $c \cdot \log n$ for some $c > 0$ (e.g., SET COVER [13]). For problems in Class III, it is quasi-NP-hard to achieve a $2^{\log^{1-\gamma} n}$ factor 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, 9, 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 [38], 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 [31]. There exists a $\ln n$ lower bound for approximating the SET COVER problem [13], however the RECTANGLE COVER problem, covering a set of axis-aligned rectangles with minimum number of points, has no such inapproximability results [35].

To the best of our knowledge, the geometric results in Section 3 provide the strongest inapproximability bounds shown for a natural, combinatorial, geometric problem. Similar lower bounds have been shown for the NEAREST LATTICE VECTOR problem [3]; however, this problem is not combinatorial, and although it shares the same lower bound as our problem, we cannot directly relate the hardness of the two problems.

3 The DISKS Problem

We consider an assembly consisting solely of disks of unit radius, whose centers lie on a polynomial-size grid in the plane. We allow disks to be removed with collision-free translations to infinity. Our goal is to remove a key disk, while minimizing the total number of disks which are removed.

Since we remove the disks one at a time, we can view an instance of the DISKS problem 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.

3.1 Scheduling with AND/OR Precedence Constraints

A complication in our formulation as a scheduling problem is that the 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 distinquishes this problem from more traditional scheduling is the form of the precedence constraints. Traditionally, a task may only have AND precedence constraints, in that it has an associated set of tasks all of which must be scheduled before that task. Unfortunately this is not the case in our assembly sequencing problem. In a single direction, if a part is to be removed, then indeed there is a clear set of parts which block that operation and thus must be removed earlier. However, we may choose to remove that same part in some other direction, in which case a different set of constraints apply.

Instead, our problem can be viewed as a special case 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 noting 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.

A model for scheduling with AND/OR precedence constraints has been studied earlier [15, 16], but with one key difference. The precedence constraints for an instance can be represented as a directed graph, with each node additionally tagged as either an AND-node or an OR-node; in this previous work, they assume that there is no cycle in this precedence graph, as that would make the problem infeasible. With AND/OR constraints, this is no longer a necessary condition for the existence of a valid solution, and in fact cycles will often exist as it may be the case that part A blocks B in one direction, B blocks C in another, and C blocks A in a third. For this reason, we make no a priori assumptions about the structure of the precedence relations. Gillies and Lin [15, 16] prove the NP-hardness of many variants of the problem, however they do not consider the approximability of the hard problems.

Notation and Definitions. 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, and has an associated set of tasks, P_i. 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. We define the OR-*degree* of an instance as the maximum sized $|P_j|$ over all OR-tasks t_j; similarly, the AND-*degree* of an instance is the maximum sized $|P_i|$ over all AND-tasks t_i. The constraints can be represented by a precedence graph, with a node for each task, t_i, and a directed edge from t_j to t_i whenever $t_j \in P_i$. A *leaf-task* is one with $P_i = \emptyset$, and thus it 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 (as in [15, 16]). We say an instance of AND/OR scheduling has *internal-tree precedence constraints* if there are no cycles, and additionally if the only nodes with more than one outgoing edge are leaf nodes.

Theorem 1 *It is quasi-NP-hard to approximate the number of* leaves *scheduled in an instance of* AND/OR *scheduling with* internal-tree *precedence constraints, to within a factor of* $2^{\log^{1-\gamma} n}$ *for any* $\gamma > 0$. *This remains true even if both the* AND-*degree and* OR-*degree are bounded by two.*

The proof is deferred to Section 4.

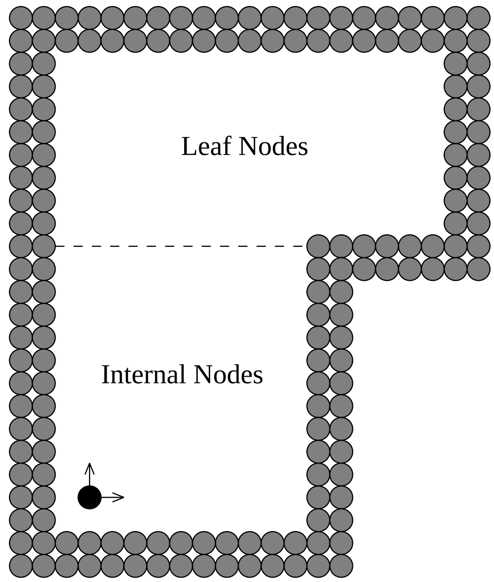

Figure 1: Overview of DISKS construction

3.2 Inapproximability of the DISKS Problem

At this point, we are able to establish the hardness result for DISKS problem, by geometrically realizing a suitably hard instance of AND/OR scheduling from Theorem 1.

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 sketch: Given a hard instance from Theorem 1 with OR-degree bounded by two (we do not require such a bound on the AND-degree), we construct an instance of the DISKS problem. We assume, without loss of generality, that OR-nodes rely only on internal nodes.

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 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, as needed, which allow a clear path out for some disks. We consider our main working area as two sections, one for the mechanism involving the interior nodes, and the second section for the leaf node mechanisms. The overview of the construction is given in Figure 1.

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 such node, T_i, we create a disk, D_i, centered at $(6i, 6i)$. We give each such disk an "escape route" to the North by creating a hole in the above wall. For an OR-disk, we create an additional passage to the East.

Finally, 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 $(6i + 1, 6k - 1)$, and force this disk to exit to the East. For an OR-node, T_i, which depends on 2 nodes, T_k and T_l, we create

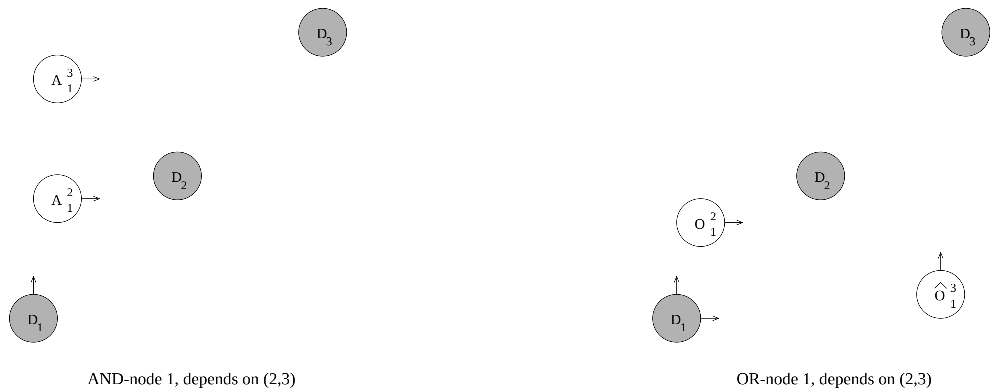

Figure 2: Internal Node Mechanisms

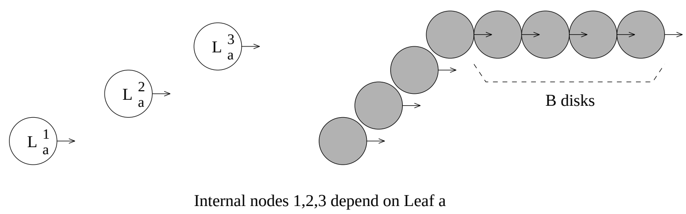

Figure 3: Leaf Node Mechanism

two new disks, O_i^k located at $(6i+1, 6k-1)$ which we force East, and $\hat{O}_i^l$ located at $(6l-1, 6i+1)$ which we force 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 (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 we force East. 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.

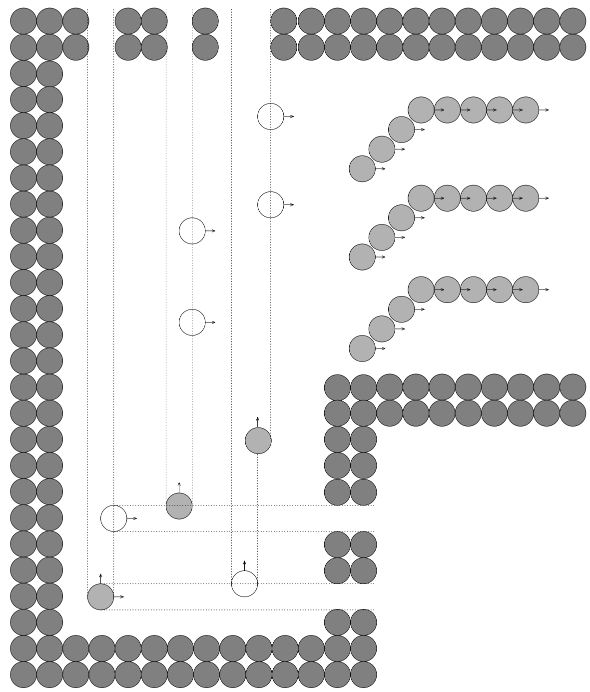

Figure 4: A Complete DISKS Construction

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, and 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, there is no need to force a restriction to linear moves, since moves which remove a group of disks at once could be replaced by a set of linear moves.

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 wall can be extended to block any useful motions in other dimensions, while still using polynomially many disks. ▮

4 Inapproximability of AND/OR Scheduling

We begin by considering the AND/OR scheduling problem when restricted to *internal-tree* precedence constraints. We will look at the problem where we only charge an algorithm for the *leaves* that it schedules[3]. We show the inapproximability of this problem by showing that the LABELCOVER$_{\min}$ problem[3, 4] is a special case. Following this, we show that the lower bound applies even when we

[3]The internal-tree defines a monotone, boolean function 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. Therefore, our results also prove the inapproximability of this problem on monotone boolean formulae. We are unaware of any previous results for this approximation problem.

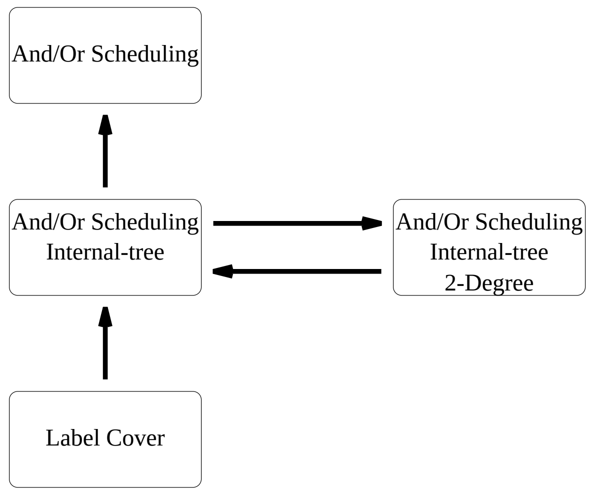

Figure 5: Reductions between variants of AND/OR scheduling

further restrict the AND/OR problem to have degree bounded by two. Finally, we convert this bound on the number of leaves scheduled, into a bound on the total number of scheduled nodes for the general AND/OR scheduling problem. An overview of the reductions is given in Figure 5.

Proof sketch for Theorem 1: 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.

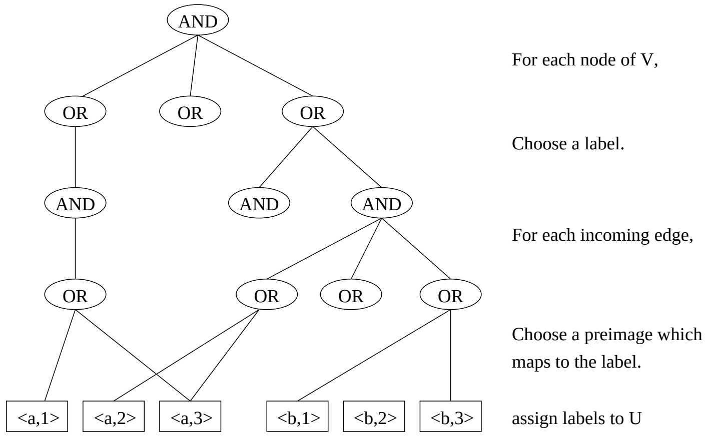

Figure 6: LabelCover as and/or scheduling with internal-tree precedence

This completes the construction. It can be seen that there is a one-to-one correspondence between valid labeling in the $\text{LabelCover}_{\min}$ 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 $\text{LabelCover}_{\min}$ instance (namely, in $|U|$, $|V|$ and N).

Combining this with the result of [4] which proves a similar lower bound for the approximability of $\text{LabelCover}_{\min}$, we get that it is quasi-NP-hard to achieve an $2^{\log^{1-\gamma} n}$ approximation for any $\gamma > 0$.

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. ∎

Theorem 3 *For the problem of minimizing the number of tasks scheduled in a general instance of* and/or *scheduling, it is quasi-NP-hard to achieve an approximation ratio of* $2^{\log^{1-\gamma} n}$ *for any* $\gamma > 0$*. This lower bound remains valid if both the* and*-degree and* or*-degree are bounded by two.*

Proof sketch: The only difficulty in switching from the measure of counting scheduled leaves to counting all scheduled nodes 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 internal-tree scheduling from Theorem 1, with 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. In Theorem 1, 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 $\gamma' > 0$. ∎

4.1 Implications of the Hardness of AND/OR Scheduling

We feel that the problem of scheduling with AND/OR precedence constraints raises several important complexity issues, of considerable interest in their own right. This form of precedence constraints is a fairly natural extension to the standard scheduling problem, yet clearly the effect of this change on the difficulty of the problem is quite dramatic. We pose a series of open directions of research related to the theory of approximability and where this problem fits in relation to several other problems.

In Figure 5, we consider several versions of this scheduling problem, giving reductions from one to another, and then we prove a lower bound of $2^{\log^{1-\gamma} n}$ against the approximability of all of these problems by showing that the easiest of these versions captures the LabelCover$_{\min}$ problem as a special case. It is open to determine a separation between any of the steps of the series of reductions. That is, the 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 [4], however it may be possible to strengthen the lower bounds for AND/OR scheduling without necessarily settling the LabelCover conjecture. Furthermore, reasoning about instances of AND/OR scheduling seems to be a bit more intuitive then about instances of a problem such as LabelCover.

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 loss of generality, we can assume that the root of our tree is an AND-node. Without a bound on the in-degree of the internal nodes, we can collapse the internal nodes into alternating levels of AND-nodes followed by levels of OR-nodes, eventually followed by a single level of leaves. Now, we can consider the complexity of the problem based on the number of alternating levels. If we consider one full alternation, that is an AND-node at the root, followed by a level of OR-nodes, followed by the level of leaves, this problem is exactly equivalent to the Set Cover problem, and hence lies in Class II. The AND-node requires that we cover each item in the universe, and each OR-node requires that for the given item, we pick one of the sets which covers that item. Each leaf corresponds to a ground set, and thus the number of leaves scheduled is equal to the number of sets used to cover the universe. If we look again at Figure 6, we see that as soon as we allow two full levels of alternations, this problem captures the LabelCover$_{\min}$ problem, and hence is in Class III. However it is not at all clear that this problem is equivalent to LabelCover as we do not know whether an instance of this restricted AND/OR scheduling can be translated into a LabelCover instance. 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 these versions? What if no constraints are placed on the structure of the precedence graph?

The answer for some of these questions may come from research in the study of monotone boolean formulae. As we mentioned earlier, the internal-tree precedences exactly defines a monotone boolean function on the leaves, where the goal is to satisfy the function using the minimum number of ones. It is clear than an arbitrarily complex formula on n leaves can be collapsed into an AND/OR tree with a single alternating level, where the top choice is of picking one of the satisfying assignments, and for each satisfying assignment, you must schedule all of the leaves which correspond to variables set to one. The problem here is that the number of internal nodes in this representation is no longer polynomial in the number of leaves, and this condition was necessary for our reductions. There is a wealth of research related to monotone formulae and circuits in this respect [32, 39, 49], however it is open to strengthen any of our inapproximability results for AND/OR scheduling.

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] B. C. et al. Applications challenges to computational geometry. Technical Report TR-521-96, Princeton University, 1996.

[12] T. D. Fazio and D. Whitney. Simplified generation of all mechanical assembly sequences. *IEEE Trans. on Robotics and Automation*, 3(6):640–658, 1987.

[13] U. Feige. A threshold of $\ln n$ for approximating set cover. In *Proc. 28th ACM Symp. Theory Comput.*, pages 314–318, 1996.

[14] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, NY, 1979.

[15] D. Gillies. *Algorithms to schedule tasks with AND/OR precedence constraints.* Ph.D. thesis, University of Illinois, Urbana, IL, 1991.

[16] D. Gillies and J. Liu. Scheduling tasks with AND/OR precedence constraints. *SIAM J. Comput.*, 24(4):797–810, 1995.

[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. Håstad. 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] 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.

[30] 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.

[31] S. Khanna and R. Motwani. Towards a syntactic characterization of PTAS. In *Proc. 28th ACM Symp. Theory Comput.*, pages 329–337, 1996.

[32] M. Klawe, W. Paul, N. Pippenger, and M. Yannakakis. On monotone formulae with restricted depth. In *Proc. 16th ACM Symp. Theory Comp.*, pages 539–550, 1984.

[33] S. Krishnan and A. Sanderson. Path planning algorithms for assembly sequence planning. In *Proc. Int. Symp. on Intelligent Robotics*, pages 428–439, 1991.

[34] S. Lee and Y. Shin. Assembly planning based on geometric reasoning. *Computers and Graphics*, 14(2):237–250, 1990.

[35] R. Motwani. Approximation algorithms. Stanford Technical Report STAN-CS-92-1435, 1992.

[36] B. Natarajan. On planning assemblies. In *Proc. 4th ACM Symp. on Computational Geometry*, pages 299–308, 1988.

[37] C. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *J. Comput. Systems Sci.*, 43(3):425–440, 1991.

[38] C. Papadimitriou and M. Yannakakis. The traveling salesman problem with distances one and two. *Mathematics of Operations Research*, 18(1):1–11, 1993.

[39] R. Raz and A. Wigderson. Monotone circuits for matching require linear depth. *J. ACM*, 39(3):736–744, 1992.

[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, and T. Lozano-Pérez. Two-handed assembly sequencing. Stanford Technical Report STAN-CS-93-1478, 1993.

[45] R. Wilson and J.-C. Latombe. Geometric reasoning about mechanical assembly. *Artificial Intelligence*, 71(2):371–396, 1994.

[46] R. Wilson and J. Rit. Maintaining geometric dependencies in and assembly planner. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 890–895, 1990.

[47] J. Wolter. *On the Automatic Generation of Plans for Mechanical Assembly.* Ph.D. thesis, University of Michigan, 1988.

[48] T. Woo and D. Dutta. Automatic disassembly and total ordering in three dimensions. *J. Engineering for Industry*, 113(2):207–213, 1991.

[49] A. Yao. A lower bound for the monotone depth of connectivity. In *Proc. 35th Symp. on Found. Comput. Sci.*, pages 302–308, 1994.