Non-Approximability of Precedence-Constrained Sequencing to Minimize Setups

Craig A. Tovey *

June 26, 2002; Revised August 3, 2003

Abstract

It is a basic scheduling problem to sequence a set of precedence-constrained tasks to minimize the number of setups, where the tasks are partitioned into classes that require the same setup. We prove a conjecture in [4, 5] that no polynomial-time algorithm for this problem has constant worst-case performance ratio unless $P = NP$. A very simple algorithm has performance ratio $\sqrt{n}$.
Keywords: computational complexity, scheduling, approximability, setup, precedence constraint.

*School of ISyE and College of Computing, Georgia Tech, Atlanta, Georgia 30332-0280. The author was supported in part by NSF grant DDM-9215467.

1 Motivation and Statement of Results

The following process planning problem is introduced by Lofgren in [4, ch. VI] and by Lofgren, McGinnis, and the author in [5].

Definition 1. *PCCS: Precedence Constrained Class Sequencing.* A set of operations must be performed sequentially, subject to an acyclic directed graph of precedence constraints. The operations are partitioned into classes. A setup is required between consecutively sequenced operations from different classes, but not within a class. The problem is to sequence the operations so as to minimize the number of setups, while obeying the precedence constraints.

If there are no classes then the problem would typically be to minimize the latest completion time, assuming unit processing time for each operation. This problem is trivial for 1 processor, polynomially solvable for 2 processors [1] or for tree precedence graphs [3], open for fixed $k \geq 3$ processors [2], and NP-complete in general [8], even with a deadline of 3 and no precedence chains of length more than 1 [7].

PCCS is a fundamental scheduling problem in systems where processors have the flexibility to perform more than one operation. In [5] PCCS is illustrated in the context of circuit card assembly. An operation is the insertion of an electronic component on a card at one of the assembly stations; each component required by a card has been assigned to one assembly station; two insertions are in the same class if they are to be performed at the same assembly station; a setup involves traveling from one station to another. In other contexts, different classes of operations might require different configurations on the same processor, and a setup might involve changes rather than travel.

NP-completeness of PCCS is shown in [5], and also several natural heuristics, including all critical path rules, are shown to have non-constant worst-case performance ratios. These results are followed by a theorem and a conjecture.

Theorem 1. [5, Theorem 4.3] If there exists a polynomial time algorithm for PCCS with constant worst-case performance ratio, then there exists a polynomial approximation scheme for the problem.

Conjecture 1. [4, 5] No polynomial time algorithm for PCCS with constant worst-case ratio exists unless $P = NP$.

This paper proves (Theorem 3) that a known MAX-SNP-Complete problem L-reduces to PCCS, and therefore PCCS has no polynomial approximation scheme. Combined with Theorem 1, this result proves Conjecture 1. Thus, though PCCS is a basic process planning problem, it is difficult to solve even approximately.

2 Proof of the conjecture

First we give some terminology. An instance of PCCS consists of a directed acyclic graph and a partition of the nodes of the graph into classes. Instead of thinking of a solution as a sequence of nodes, let a solution be a sequence of classes which are applied to clear away nodes from the graph. When a class is applied, all the nodes in that class are cleared, except those that have a uncleared predecessor that is not in the same class. A feasible solution consists of a sequence of classes which, when applied, clears

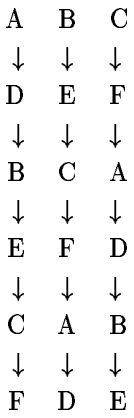

Figure 1: Gadget for Clauses

the entire graph. The value $v(s)$ of a solution s to an instance of PCCS is thus simply the number of steps in the sequence, $|s|$. It is easy to show that the schedules we get from such class sequences are dominant with respect to $v(s)$.

Next we introduce a gadget, which is derived from [5, Figure 2].There are 18 nodes, 6 classes denoted A,B,...,F, and a precedence graph of 3 paths arranged in 6 rows, as shown in Figure 1.

Lemma 2. The gadget graph in Figure 1 can be optimally cleared in 10 steps. If one or more nodes in the gadget's top row are deleted, the resulting graph can be optimally cleared in 9 steps.

Proof. First, the sequence ADBECFADBE clears the gadget in 10 steps. Second, if the A in the top row is removed, then omitting the first A in the sequence just given would clear the gadget in 9 steps. By symmetry 9 steps would suffice if a different node from the top row were removed. Third, it takes at least 6 steps for all the 2nd elements in the three paths to be cleared. After the last of the 2nd elements is cleared, the path below it is length 4 and requires at least 4 more steps. Hence the gadget cannot be cleared in fewer than 6+4=10 steps. Fourth, if the top row of the gadget is removed, a similar argument shows that at least 6+3 steps are required to clear. ■

The main idea of the gadget is that it is good to lop off one of the elements from the top row, but it does not help to lop off more than one.

Now we define 3MAX3SAT (called "3-OCCURENCE MAX3SAT" in [6]), the known MAXSNP-complete problem that we will L-reduce to PCCS. An instance q of 3MAX3SAT is a set of boolean variables $x_1, \ldots, x_n$ and clauses $c_1, \ldots, c_m$ where each clause contains exactly 3 literals. A literal is a variable appearing uncomplemented as x_i or complemented as $\bar{x}_i$. Each variable appears in at most 3 clauses, and can be assumed to appear in at least 2 clauses, whence $2n \leq 3m \leq 3n$. A clause is true iff at least one of its literals is true. The value $v(a, q)$ of a truth assignment a is the number of clauses that are true with respect to a. $OPT(q)$ is the maximum over a of $v(a, q)$.

We L-reduce 3MAX3SAT to PCCS to show the latter cannot be approximated to arbitrary accuracy. To be specific, logspace functions R and T comprise an L-reduction [6, p.309]) of 3MAX3SAT to PCCS if there exist positive constants α, β such that

- R transforms instances q of 3MAX3SAT to instances $R(q)$ of PCCS;
- for all instances q of 3MAX3SAT, $OPT(R(q)) \leq \alpha OPT(q)$;
- for an instance q and a feasible solution s to $R(q)$, $T(s)$ is a feasible solution to q such that $|OPT(q) - v(T(s), q)| \leq \beta |s| - OPT(R(q))|$.

Theorem 3. 3MAX3SAT L-reduces to PCCS.

Proof. *The mapping R.* We construct the instance of PCCS from an instance of 3MAX3SAT. The node classes in the graph are

- $X_{it}, \bar{X}_{it} : i = 1 \ldots, n; t = 1, 2, 3$. These represent 3 copies of the variables and their complements. The 3 copies balance the 3 occurrences of the variable in clauses. It will not be worth "cheating" a variable at cost 3 to gain a benefit of 3 on the clauses. (That is, it will turn out that had we only one copy for a variable x_i it would be possible in effect to set both x_i = TRUE and x_i = FALSE at a cost of 1 in the PCCS instance, and thereby potentially satisfy more than one additional clause. With 3 copies per variable, the cost outweighs the potential benefit.)
- $C_{ij}, D_{ij} : (i, j) : x_i \in c_j$ or $\bar{x}_i \in c_j; j = 1, \ldots, m$. For each occurence of a variable x_i (or its complement) in clause c_j there is a class C_{ij}. There is also a dummy class D_{ij}. Exception: if a literal occurs twice in the same clause, create two classes C_{ij} and $C_{i'j}$, and similarly with the D's, one for each occurence.
- A single node class W.

The graph itself consists of 3 parts (see Figure 2). The first part of the graph functions as a schedule controller. Its nodes are in 4 levels. There is a precedence arc from every node in every level to every node in the level below. There are no precedence relations between nodes in the same level.

Level I One node of each class $X_{it}, \bar{X}_{it}; \forall i, t$

Level II One node of each class C_{ij}

Level III One node of each class $X_{it}, \bar{X}_{it}; \forall i, t$

Level IV A single node W.

This controller will force the solution to occur in 5 phases. In phase I, the solution must clear level I by scheduling all the copies of the variables and complements, in some order. Phase II clears all of the C_{ij} of level II; phase III clears all the level III copies of the variables and their complements; phase IV clears the node W. Finally, phase V clears the clause gadgets.

The second part of the graph consists of the variable gadgets (see Figure 3). There are n of these 12-node components, each representing a variable x_i. For every $i = 1 \ldots, n; t = 1, 2, 3$ the component consists of two rows of six nodes each. The upper row contains nodes from classes $X_{it} : t = 1, 2, 3$ followed by $\bar{X}_{it} : t = 1, 2, 3$ The lower row contains nodes from classes $\bar{X}_{it} : t = 1, 2, 3$ followed by $X_{it} : t = 1, 2, 3$ There are 6 precedence arcs, one from each node in the upper row to the node directly below it.

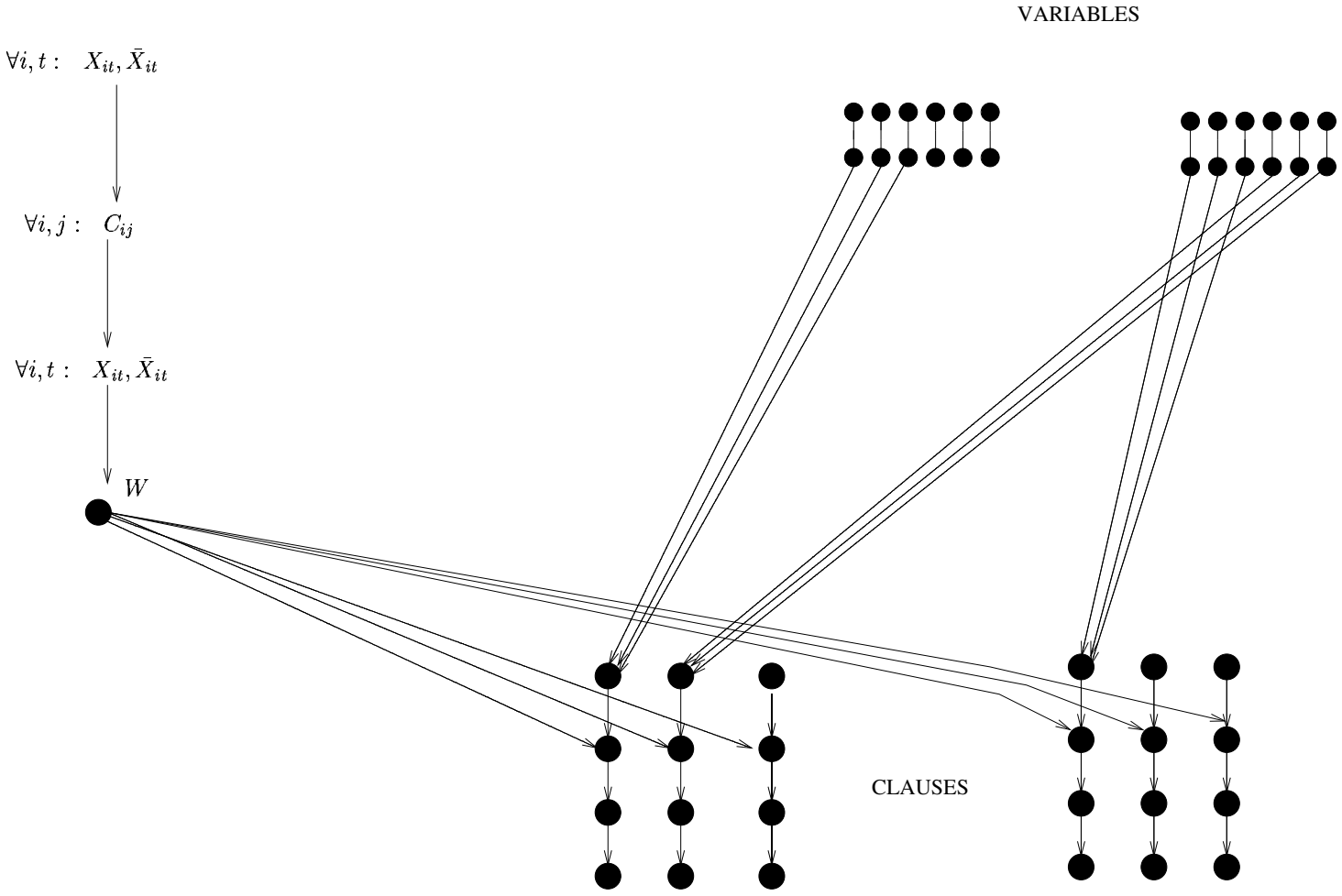

Figure 2: The three parts of the graph are the schedule controller, the variable gadgets, and the clause gadgets.

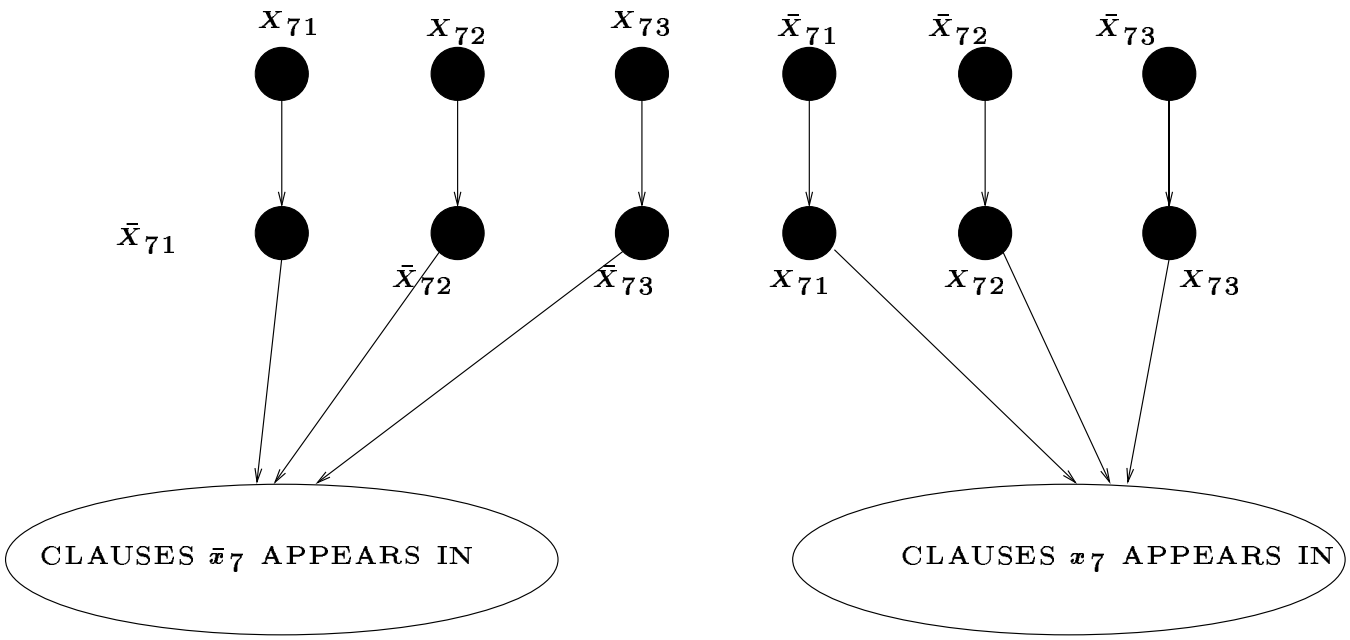

Figure 3: Variable gadget for x_7. Phase I will free up clauses that x_7 is in if $\bar{x}_{7t}$ precedes x_{7t}. The reverse order frees up clauses that $\bar{x}_7$ appears in.

If all three copies of the variables and their complements are sequenced, setting x_i = TRUE (resp. FALSE) corresponds to X_{it} occuring later (resp. earlier) than $\bar{X}_{it}$ in the sequence, for all t. For example, in Figure 3, which depicts the variable gadget for x_7, phase I will free up clauses containing the literal x_7 if the class sequence is $\bar{X}_{71}, \bar{X}_{72}, \bar{X}_{73}, X_{71}, X_{72}, X_{73}$. Phase I will free up clauses containing the literal $\bar{x}_7$ if the class sequence is $X_{71}, \bar{X}_{71}, X_{72}, \bar{X}_{72}, X_{73}, \bar{X}_{73}$.

The third part of the graph consists of the clause gadgets. See Figure 4 for an example. These have the internal form depicted in Figure 1 and analyzed in Lemma 2, where A, B, and C denote the three variable occurences C_{ij} in the clause, and D, E and F denote the three dummy classes D_{ij}. Besides the precedences within the columns already stated, there are precedences from the other two parts of the graph:

- An arc from W to every D_{ij} node in the second row of each clause gadget.
- If $x_i \in c_j$, an arc from each $X_{it} : t = 1, 2, 3$ in the second row of the x_i variable gadget to C_{ij} in the first row of the clause gadget c_j. Similarly, if $\bar{x}_i \in c_j$, an arc from the second row $\bar{X}_{it} : t = 1, 2, 3$ to the first row C_{ij}.

Figure 4 helps us see how satisfying a clause corresponds to getting a shorter feasible schedule in the constructed graph. If clause $j = 4$ is $\{x_5, \bar{x}_6, \bar{x}_7\}$, then if for all t class $\bar{x}_{5t}$ precedes x_{5t}, or class x_{6t} precedes $\bar{x}_{6t}$, or class x_{7t} precedes $\bar{x}_{7t}$, at least one node in the top row of the $j = 4$ clause gadget will clear during phase II. The node W prevents any nodes in the 2nd row from clearing until phase V. Therefore, by Lemma 2, during phase V the clause gadget will require 9 steps to clear. If none of these three precedences occurs, then the $j = 4$ clause gadget is completely intact at the end of phase II, and will require 10 steps to clear in phase V.

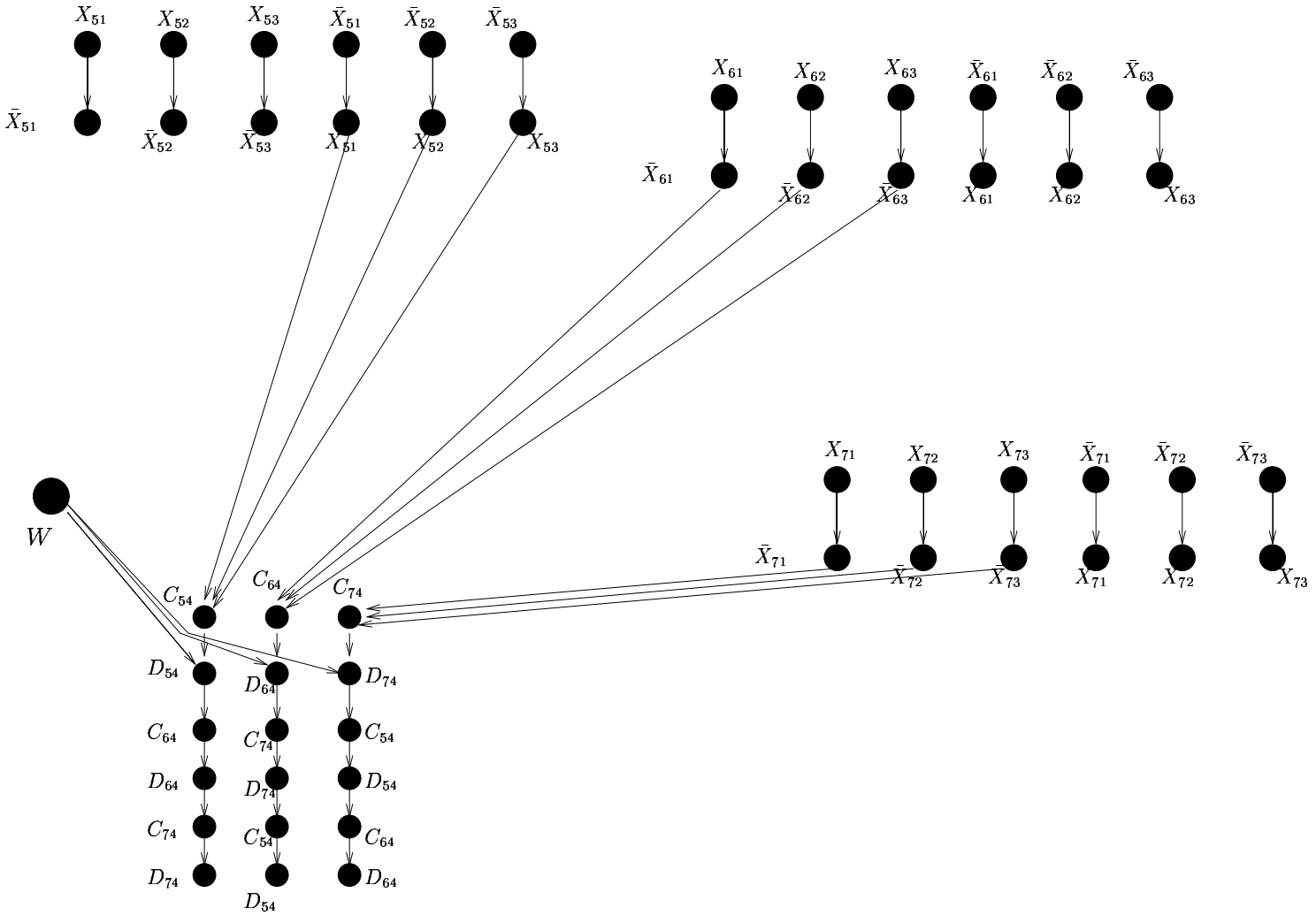

Figure 4: Clause gadget for clause $j = 4$ consisting of $\{x_5, \bar{x}_6, \bar{x}_7\}$.

Lemma 4. Let a be a truth assignment satisfying $v(a,q)$ clauses of the instance q on n variables and m clauses. Then there exists a corresponding solution $f(a,q)$ that is feasible for $R(q)$, of value $|f(a,q)| = 12n + 13m + 1 - v(a,q)$. Moreover, $f()$ is logspace computable.

Proof: We construct $f(a,q)$ in five phases, corresponding to the five phases of a solution. Phase I: For all $i = 1,\ldots,n$ and $t = 1,2,3$ put $\bar{X}_{it}$ followed by X_{it} if x_i = TRUE in the assignment a, and in the reverse order if x_i = FALSE. In the variables gadgets, this clears all the X_{it} corresponding to TRUE variables, and all the $\bar{X}_{it}$ corresponding to FALSE variables. In the control part of the graph, this clears all of level I. This phase costs $6n$.

Phase II: Put $C_{ij}\forall i,j$. This clears all of level II in the control part of the graph, and clears every node in the top row of each clause gadget that corresponds to a literal made true by the assignment a. This phase costs $3m$.

Phase III: Put $\bar{X}_{it}$ and X_{it} for all $i = 1,\ldots,n$ and $t = 1,2,3$. This clears all of level III in the control part of the graph and clears all remaining nodes in the variables gadgets. This phase costs $6n$.

Phase IV: Put W in the sequence. At this point all remaining nodes are in clause gadgets.

Phase V: Sequence the nodes in each clause gadget according to Lemma 2. Each clause costs either 9 or 10 to clear, depending on whether any of its nodes were cleared in step II or not. The cost for this phase is therefore $10m - v(a,q)$.

Each phase is logspace computable, and the total cost of the schedule constructed is $12n+1+13m-v(a,q)$. ■

For all q there are a for which $v(a,q) > 0$. By Lemma 4, there are solutions to $R(q)$ of length $12n+1+13m-v(a,q) \leq 12n+13m \leq 31m$. Therefore, for an instance q with m clauses, $OPT(R(q)) \leq 31m$.

For any instance q of 3MAX3SAT with m clauses, $OPT(q) \geq m/2$ because any truth assignment a and its complement $\bar{a}$ has $v(a,q) + v(\bar{a},q) \geq m$.

Combining the upper bound on $OPT(R(q))$ and the lower bound on $OPT(q)$ we get

$$OPT(R(q)) \leq 31m \leq 62OPT(q).$$

Thus the first inequality of an L-reduction is satisfied by our mapping R (with $\alpha = 62$).

The mapping T. Now we develop the function T which transforms every feasible solution s for $R(q)$ to a feasible solution for q.

Until now, we have applied the idea of phases only to our own constructed schedules. We now make a definition that separates any feasible schedule into five phases. If s is a feasible solution to $R(q)$ it must clear all of level I of the control portion of the graph. Define s_I, the first phase of s, to be from the beginning of s until level I is cleared. The first phase has length $|s_I| \geq 6n$ because it must contain (at least one) $\bar{X}_{it}$ and X_{it} for all $i = 1,\ldots,n$ and $t = 1,2,3$. Let the "extra" part of phase I, e_I, denote the subsequence of s_I that remains if the first occurences of $\bar{X}_{it}$ and $X_{it}\forall i\forall t$ are deleted. Thus $|s_I| = 6n + |e_I|$. Note e_I can be empty.

Similarly, define phases II, III, and IV as the portion of s from the end of the previous phase until the corresponding level is cleared. The precedence constraints between levels in the control force

$|s_{II}| \geq 3m; |s_{III}| \geq 6n$, and $|s_{IV}| \geq 1$. Define e_{II}, e_{III}, e_{IV} similarly as the extra portions of these phases.

Define phase V as the portion of s from the end of phase IV until the end of s. The precedence arcs from W, the definition of phase IV, together with Lemma 2 imply that (if s is feasible) the length of phase V is at least $9m$ plus the number of clause gadgets untouched at the end of phase IV.

Phases I and III must each contain at least one each of X_{it} and $\bar{X}_{it}$ for all i, t. Therefore the node gadgets are all completely cleared by the end of phase III.

We now define T. Let q be an instance of 3MAX3SAT and let s be a feasible solution to $R(q)$. Fix i. If x_i never appears complemented (resp. uncomplemented) in a clause, trivially set $x_i =$ TRUE (resp. FALSE). Otherwise, consider the 6 non-extra $X_{it}, \bar{X}_{it}$ terms of s_I. That is, consider the subsequence of s_I consisting of the first occurence of each of these 6 terms. If the X_{it} term occurs later (resp. earlier) than the $\bar{X}_{it}$ term for two or three values of t then assign $x_i =$ TRUE (resp. FALSE). The key idea is that if $x_i =$ TRUE, then e_I would have to contain at least two $\bar{X}_{it}$ terms to clear all the $\bar{X}_{it}$ nodes in the variables gadget during phase I, because at least two $\bar{X}_{it}$ nodes must be cleared.

Lemma 5. Let s be a feasible solution to $R(q)$. Set $s' = f(T(s))$, where $f()$ is defined in Lemma 4. Then s' is feasible and $|s'| \leq |s|$.

Proof. Feasibility follows from Lemma 4. By the same lemma, $|s'| = 12n + 13m - v(T(s), q)$. Let $V(s)$ denote the number of clause gadgets in $R(q)$ that have already been partly cleared at the start of phase V. By Lemma 2, $|s| \geq |s_I| + |s_{II}| + |s_{III}| + |s_{IV}| + 10m - V(s)$. Therefore it suffices to prove that $V(s) - v(T(s), q) \leq |e_I| + |e_{II}| + |e_{III}| + |e_{IV}|$. To prove this it suffices to show that there are at least as many extra terms in $e_I \cup e_{II} \cup e_{III} \cup e_{IV}$, as there are clauses c_j that are not satisfied by $T(s)$ but whose gadgets are partly cleared at the start of phase V.

Let c_j be such a clause; we will assign a distinct extra term to it. By symmetry on complements, suppose without loss of generality the literal $x_i \in c_j$ is cleared by s before phase V. We know $T(s)$ assigns $x_i =$ FALSE (else $T(s)$ would satisfy c_j), so the literal $\bar{x}_i$ must appear in at least one clause (else T would trivially have assigned $x_i =$ TRUE). The variable has at most 3 occurrences, hence the literal x_i appears in at most 1 clause of the desired type other than c_j. Also, by construction of T, X_{it} precedes $\bar{X}_{it}$ for at least 2 values of t, among the 6 non-extra terms of s_I.

Were it not for the terms in $e_I, \ldots, e_{IV}$, at least 2 X_{it} nodes in the x_i variable gadget would not be cleared at the end of phase I, and C_{ij} in the c_j gadget would not clear in phase II of s. The easy way for C_{ij} to be cleared is by an extra C_{ij} term late in e_{III} (after X_{ij}) or in e_{IV}. Such an extra C_{ij} term cannot help clear any other clause gadget, hence it can be assigned to c_j. The more complicated way for C_{ij} to be cleared is by extra X_{it} terms for at least 2 values of t, in e_I or early in e_{II} (prior to C_{ij}). Assign 1 of these extra X_{it} terms to clause c_j. If the literal x_i appears in another clause (it can appear in at most 1 other), there is another extra X_{it} term available to be assigned to it. Therefore each clause can be assigned a distinct extra term, proving the lemma. ■

Let s be feasible for $R(q)$. Let $s' = f(T(s), q)$ and let s^* be optimal for $R(q)$. Then by Lemma 5 $|s^*| \leq |s'| \leq |s|$. So

$||s'| - |s^*|| \leq ||s| - |s^*|| = ||s| - OPT(R(q))|$. Now $|s'| = 12n + 13m + 1 - v(T(s), q)$ by Lemma 4.

Apply Lemma 4 to $OPT(q)$ to get a feasible sequence of length $12n + 13m + 1 - OPT(q)$. By

optimality of s^*, $|s^*| \leq 12n + 13m + 1 - OPT(q)$. Rearranging, and substituting for $|s'|$ with the equation preceding, $OPT(q) \leq 12n + 13m + 1 - |s*| = |s'| + v(T(s), q) - |s^*|$. Rearranging again gives $OPT(q) - v(T(s), q) \leq |s'| - |s^*|$. Since q is an instance of a maximization problem and PCCS is a minimization problem, we can introduce absolute values to get $|OPT(q) - v(T(s), q)| \leq ||s'| - |s^*|| \leq ||s| - OPT(R(q))|$ as desired, with $\beta = 1$. This completes the proof that R and T L-reduce 3MAX3SAT to PCCS.

■

Corollary 6. PCCS does not have a polynomial approximation scheme unless $P = NP$.

Proof. A polynomial approximation scheme for PCCS would provide one for 3MAX3SAT via the L-reduction of Theorem 3. But no such scheme exists for 3MAX3SAT unless $P = NP$ [6]. ■

Combining Corollory 6 with Theorem 1 proves Conjecture 1.

Theorem 7. No polynomial time factor k approximation algorithm for PCCS exists, for any constant k, unless $P = NP$.

Finally we remark that there is a very simple factor $\sqrt{n}$ approximation algorithm for PCCS, where n is the number of operations. Let c be the number of (nonempty) classes. Let z^* denote the optimal schedule length. Note $z^* \geq c$. If $c \leq \sqrt{n}$ then the sequence $\{1, \ldots, c\}$ repeatedly applied will clear the graph in at most z^* repetitions, at cost at most $cz^* \leq \sqrt{n}z^*$. On the other hand, if $c > \sqrt{n}$, any feasible solution of length n has cost $n \leq \sqrt{n}c \leq \sqrt{n}z^*$. The proof of Theorem 1 [5] gives no improvement on this performance ratio. This leaves an interesting gap between $\sqrt{n}$ and Theorem 7.

References

[1] M. Fujii, T. Kasami, K. Ninamiya, "Optimal Sequencing of Two Equivalent Processors", *J. SIAM* **17** (1969), 784–789.

[2] M. Garey, D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman (1978).

[3] T.C.Hu, "Parallel Sequencing and Assembly Line Problems," *J. ORSA* **9** (1961), 841–848.

[4] C. Lofgren, "Machine Configuration of Flexible Printed Circuit Board Assembly Systems", *Ph.D. Thesis, School of ISyE, Georgia Institute of Technology*, August 1986.

[5] C. Lofgren, L. McGinnis,C.Tovey, "Routing Circuit Boards Through an Assembly Cell", *Operations Research* **39** (1991), 1012–1024.

[6] C. H. Papadimitriou, *Computational Complexity.* Addison-Wesley, 1994.

[7] C. Tovey, "A simplified anomaly and reduction for precedence constrained multiprocessor scheduling," *SIAM J. Disc. Math.* **3** (1990) 582–584.

[8] J. Ullman, "NP-complete scheduling problems," *J. Comput. System. Sci.* **10** (1975), 384–393.