STAN-CS-72-310 SEL-72-040

Anomalies in Scheduling Unit-Time Tasks

by

Marc T. Kaufman

June 1972

Technical Report No 34

This work was supported in part by the National Science Foundation
under Grant GK 23315 and by the
Joint Services Electronic Programs
U.S. Army, U.S. Navy and U.S. Air Force
under contract N-00014-67-A-0112-0044

DIGITAL SYSTEMS LABORATORY

STANFORD ELECTRONICS LABORATORIES

STANFORD UNIVERSITY • STANFORD, CALIFORNIA

STAN-CS-72-310 SEL-72-040

ANOMALIES IN SCHEDULING UNIT-TIME TASKS

by

Marc T. Kaufman

June 1972

Technical Report No. 34

DIGITAL SYSTEMS LABORATORY

Department of Electrical Engineering Department of Computer Science

Stanford University

Stanford, California

This work was supported in part by the National Science Foundation under Grant GK 23315 and by the Joint Services Electronic Programs U.S. Army, U.S. Navy and U.S. Air Force under contract N-00014-67-A-0112-0044.

ABSTRACT

In this paper we examine the problem of scheduling a set of tasks on a system with a number of identical processors. Several timing anomolies are known to exist for the general case, in which the execution time can increase when inter-task constraints are removed or processors are added. It is shown that these anomalies also exist when tasks are restricted to be of equal (unit) length. Several, increasingly restrictive, heuristic scheduling algorithms are reviewed. The "added processor" anomaly is shown to persist through all of them, though in successively weaker form.

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

In this paper, we consider the problem of scheduling computations for a system with n identical processors. The computations are represented by directed acyclic graphs. The nodes of a graph represent tasks with specified execution time. The directed arcs of a graph represent a precedence ordering of the tasks; i.e.: if there is a directed path from task T_i to task T_j, then task T_j cannot be started until task T_i is finished.

We are interested in finding algorithms that schedule the tasks on the processors such that the total execution time is minimized. Optimal solutions that do not require exponential time to compute are known for only a small class of problems; namely tree graphs in which every task is the same (unit) length [7], and directed acyclic graphs of unit length tasks when the number of processors, n, is 2 [2,3]. Much research has been directed toward finding "reasonable" heuristics which give good, though non-optimal, results [1,8,9].

Graham [4,5] has shown that the possible degree of non-optimality for a schedule is bounded by the expression $w'/w_0 \leq 1 + (n-1)/n'$, where w_0 is the total (optimal) execution time using n processors and w' is the total execution time with n' processors. For n=n', this reduces to $w'/w_0 \leq 2 - 1/n$.

We would like to know if this bound could be reduced by requiring that certain constraints be met with respect to the scheduling of the tasks and the tasks themselves. Of course, we would require that the constraints could be fulfilled in a reasonably efficient manner computationally (say, in algebraic time).

It is the purpose of this paper to show that, at least for several simple and widely used heuristics, severe degradation can still occur. Several heuristics are presented with examples of graphs that show degradation using the heuristics.

Notation and Convention

Suppose we are given a set of tasks, $\tau = \{R,S,T,\ldots\}$, a set of identical processors $P = \{p_1,p_2,\ldots,p_n\}$, and a partial ordering α on τ. In this paper, only systems with identical (unit) length tasks are considered. We can define a schedule, L, as an ordering of the tasks in τ.

We define the execution of τ according to L as follows: at any time t (beginning with t=0), we scan the list L from the beginning and mark the first n tasks encountered that are executable. A task is executable if it has not been assigned to an earlier time and if all of its predecessors, if any, have been executed at earlier times. There may be fewer than n tasks that can be marked in this way. The marked tasks are assigned to the processors $(p_1,p_2,\ldots)$ at this time and executed. When no task can be assigned, execution terminates. The total execution time, w, is the latest time at which the execution of a task in τ is terminated.

Optimizing the total execution time involves finding a schedule L_0 such that the total execution time for this schedule is minimal over all schedules for the system of tasks.

The Anomalies

Graham [4,5] has shown that the time required to process a set of tasks can increase if any of the following are done, singly or in combinations:

1. Reorder the schedule (L' = permutation of L)
2. Relax the partial order ($\alpha' \subseteq \alpha$)
3. Increase the number of processors ($n' \geq n$)

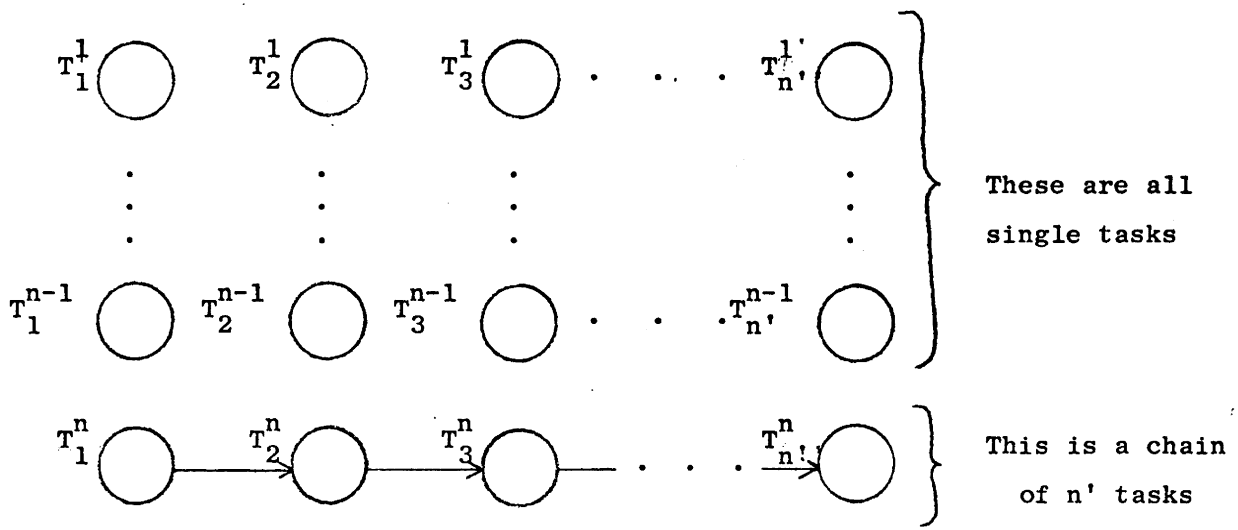
with the maximum degradation bounded by $w'/w \leq 1 + (n-1)/n'$.

These results are for general systems in which the execution times for each task could be arbitrary. As stated earlier we consider these anomalies for the case where all tasks are of unit duration.

Example 1: Varying the schedule, L, and (possibly) n.

Consider the system shown in Figure 1. If schedule L is used, with n processors, the total execution time is: w=n', If schedule L' is used, with n' processors, the total execution time increases to: w' = n-1+n'. The degradation is given by: $w'/w = 1 + \frac{n-1}{n'}$, which achieves Graham's bound.

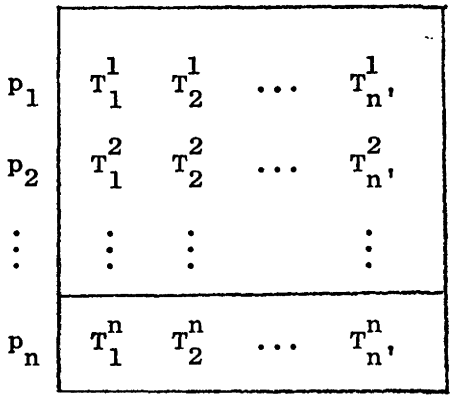
If we follow the Gantt chart for this example shown in Figure 2, we see that the apparent cause of the degradation is the early scheduling of a large number of independent tasks. This in turn forces the chain of tasks to a very late start and subsequent late termination. It will become clear that this is the mechanism behind all of the anomalies in this paper.



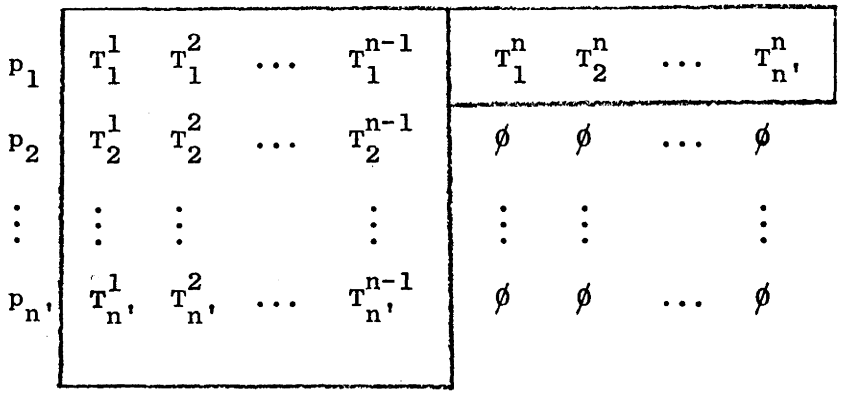
$$L = (T_1^1, \ldots, T_1^{n-1}, T_1^n, T_2^1, \ldots T_2^{n-1}, T_2^n, \ldots, T_{n'}^1, \ldots, T_{n'}^{n-1}, T_{n'}^n)$$

$$L' = (T_1^1, T_2^1, \ldots, T_{n'}^1, \ldots, T_1^{n-1}, T_2^{n-1}, \ldots, T_{n'}^{n-1}, T_1^n, T_2^n, \ldots, T_{n'}^n)$$

Fig. 1. A system which degrades when the schedule is varied.



a) Execution of Fig. 1 with schedule L



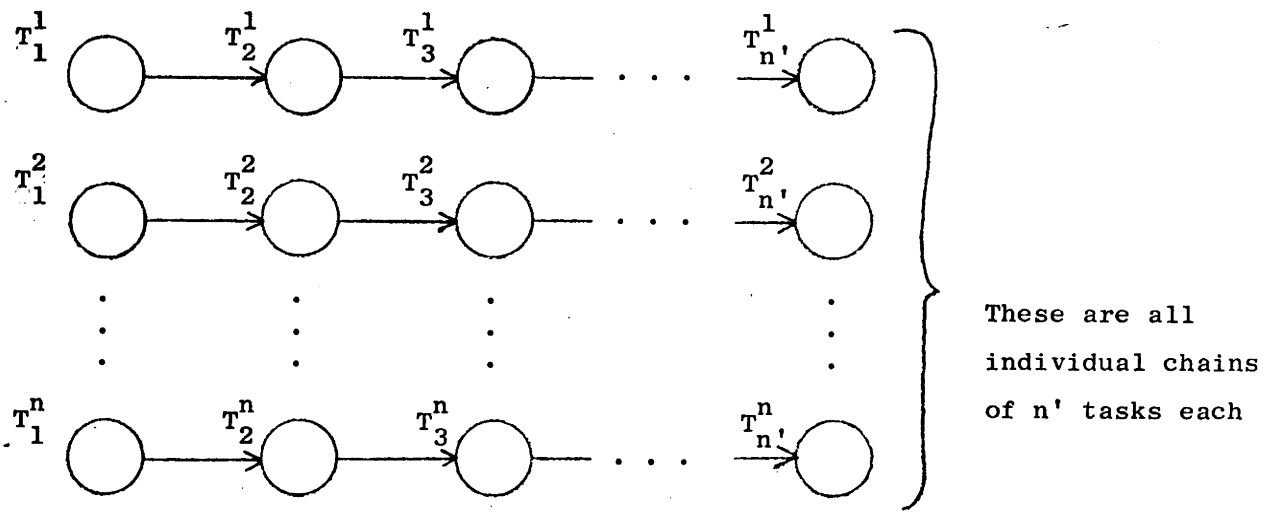
b) Execution of Fig. 1 with schedule L'

Fig. 2. Gantt charts for the system of Fig. 1.

Example 2: Relaxing the partial order, α, and (possibly) varying n.

This example is shown in Figure 3. Execution of the system presented, with n processors, yields w=n'. If the partial ordering is relaxed, by removing all constraints between tasks in the first n-1 chains (leaving chain $T_1^n \ldots T_{n'}^n$), then execution is again the Graham bound.

Examples 1 and 2 are very similar. The constraints for optimality in the first case being enforced by selection of the schedule, and in the second case by additional explicit links in the partial ordering. This type of tradeoff is generally possible when inventing such examples.



$$L = (T_1^1, T_2^1, \ldots, T_{n'}^1, T_1^2, T_2^2, \ldots, T_{n'}^2, \ldots, T_1^n, T_2^n, \ldots, T_{n'}^n)$$

Fig. 3. A system which degrades when the partial order is relaxed.

Example 3: Increasing n (only).

This type of anomaly, that due to increasing the number of available processors, is the subject of the next several examples. The best schedule would be one for which execution with n processors would always yield an optimal time (for n processors, not necessarily a global optimal time). It would certainly be disheartening to find that the total execution time may increase when more processors are used. Unfortunately, this can be the case.

Consider the system shown in Figure 4. The two constants, K and ℓ, used in the construction are arbitrary with the following restrictions:

For any n,n', select K and ℓ such that

$$K \geq 2; \quad 0 < \ell = \left\lfloor \frac{K(n-1)-n}{n'} \right\rfloor .$$

The schematic Gantt charts for execution of this system with n and n' processors are shown in Figure 5. With n processors, w=K. With n' processors, w'= K+ℓ. The ratio of these times is:

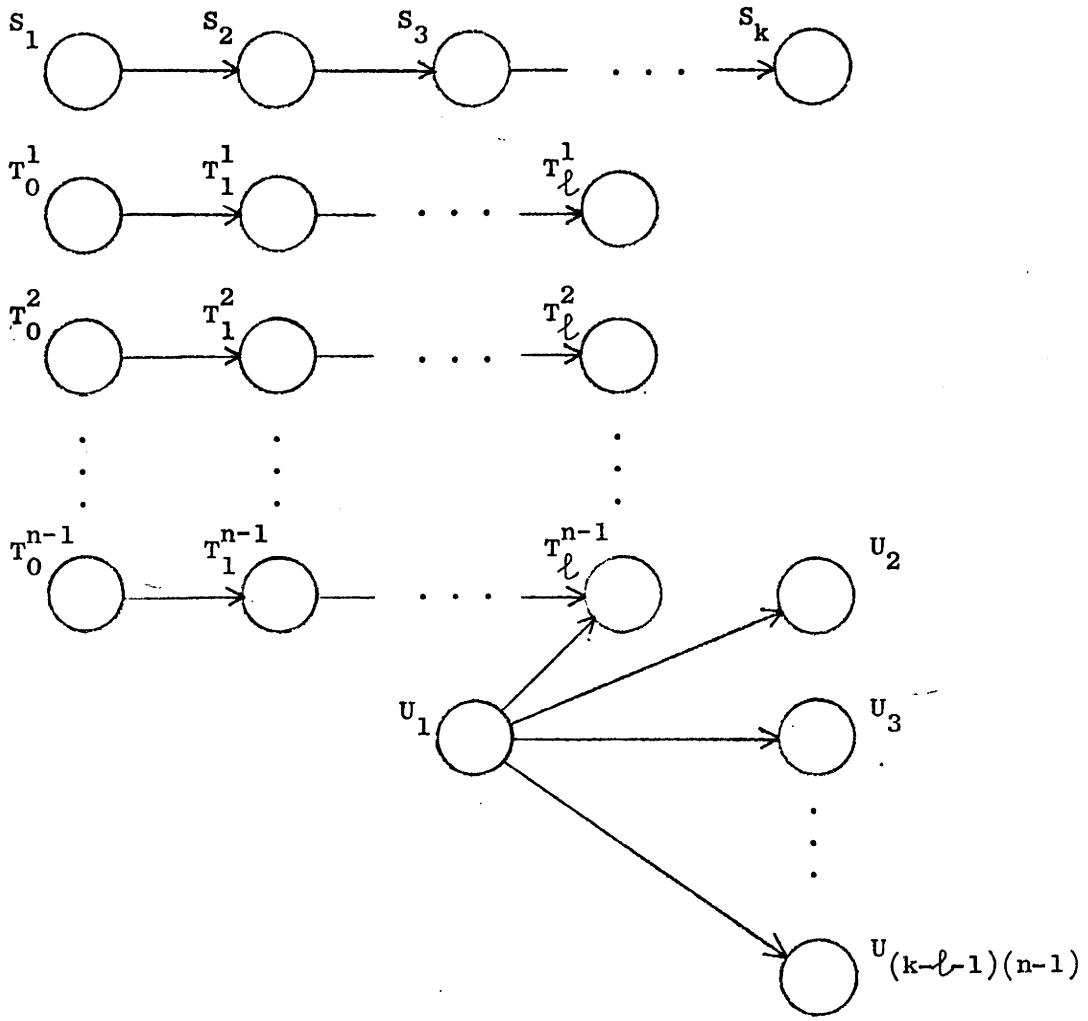
$$\begin{aligned} w'/w &= \frac{K+\ell}{K} \\ &= 1 + \frac{\ell}{K} \\ &= 1 + \frac{1}{K}\left\lfloor \frac{K(n-1)-n}{n'} \right\rfloor \\ &\leq 1 + \frac{1}{K}\left(\frac{K(n-1)-n}{n'}\right) \\ &\leq 1 + \frac{n-1}{n'} - \frac{n}{Kn'} \end{aligned}$$

We see that w'/w can be brought arbitrarily close to $1 + \frac{n-1}{n'}$ by increasing K.

Note that task precedence is not preserved in L (task precedence is preserved if, for any two tasks $R,S \in \tau$, $R < S$ implies that R preceeds S in L). In this example, nonpreservation of task precedence is what lets

us "break the barrier" when U_1 is executed, thereby letting all of the U_i come to be immediately available for assignment.

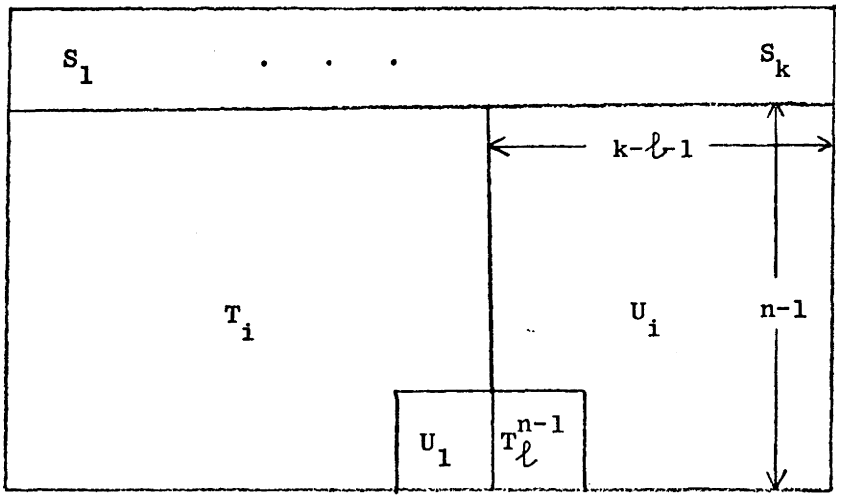
Since preservation of precedence is an easily checked property, we next look at this same anomaly, with the restriction that the schedule L be precedence preserving.



$$L = (S_1, T_0^1, T_1^1, \ldots, T_\ell^1, T_0^2, \ldots, T_\ell^2, \ldots, T_0^{n-1}, \ldots,$$

$$T_\ell^{n-1}, S_3, S_4, \ldots, S_k, U_2, U_3, \ldots, U_{(k-\ell-1)(n-1)}, S_2, U_1)$$

Fig. 4. A system which degrades with increasing n.



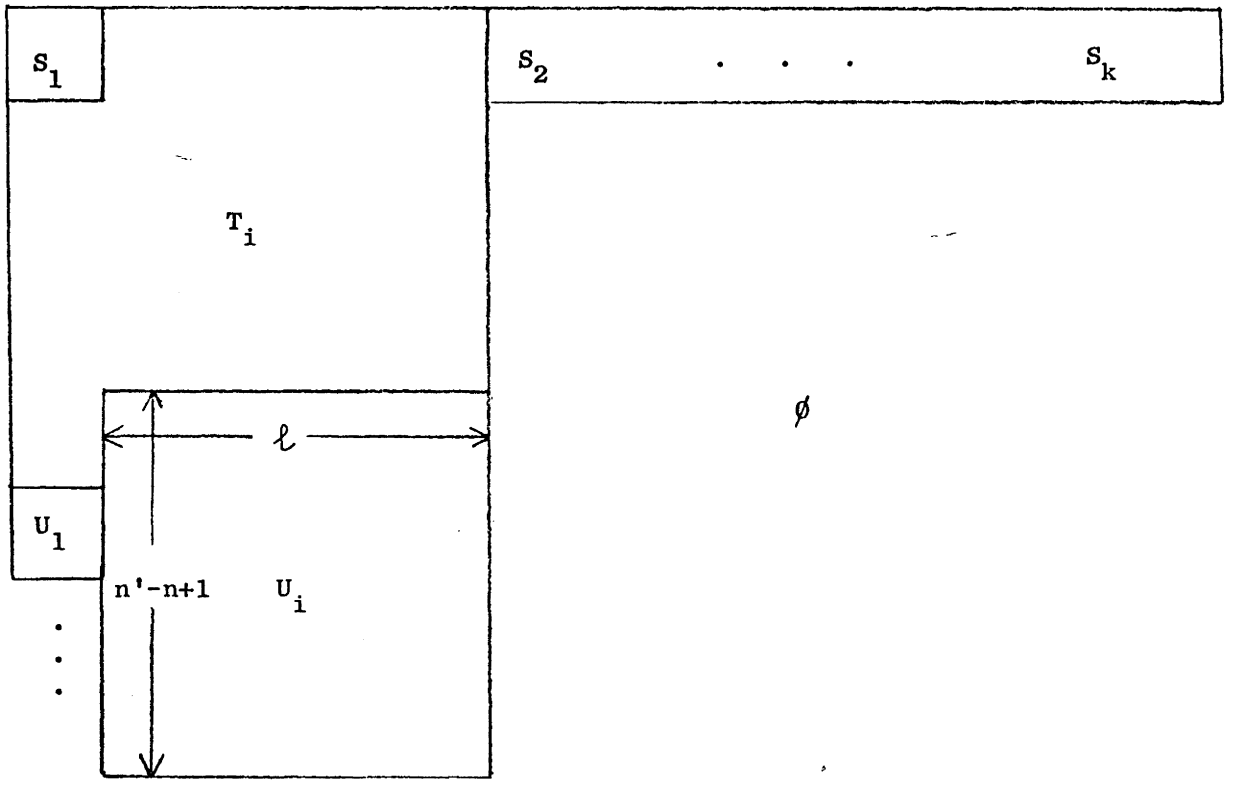


Fig. 5. Gantt charts for the system of Fig. 4.

Example 4: Increasing n (only), with a precedence-preserving list.

First let us consider the example shown in Figure 6. Execution with n=4 and n'=5 processors is given by the Gantt charts in Figure 7. Notice that, when n is increased to n', the earlier execution of T_0 allows T_1,T_2, and T_3 to be assigned with S_1 and S_2, thereby delaying R_1 and R_2.

A fully general construction is not known for this case; however, when n and n' satisfy the relations: $n' > n$, $Kn-K-1 \geq (K-1)n'$, the construction of Figure 8 may be used.

Once past the starting tasks, U_1, ..., U_n, the system consists of any number (ℓ) repetitions of the same subgraph. Execution of this network is shown by the Gantt charts in Figure 9. We see that $w'/w = \frac{(2K-1)\ell + 1}{K\ell+1}$ which, for sufficiently large ℓ, arbitrarily closely approaches $2 - \frac{1}{K}$.

The degradation is given in terms of the parameter, K. We would like to see how this compares to Graham's bound, which is in terms of n and n'.

We note that K is governed by the relation:

$$1 \leq K \leq \frac{n'-1}{n'-n+1}$$

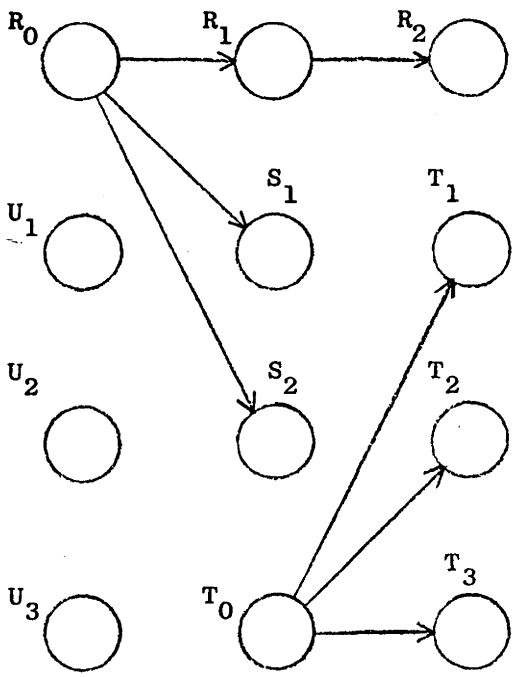
Substituting into the expression for degradation of this example we get:

$$w'/w \leq 2 - \frac{1}{K} \leq 1 + \frac{n-2}{n'-1} = 1 + \frac{n-1}{n'} - \frac{n'-n+1}{n'(n'-1)}$$

which differs from Graham's bound by a term of $O\left(\frac{1}{(n')^2}\right)$. In the specific case where n'=n+1, the expression reduces to:

$$w'/w \leq 1 + \frac{n-1}{n+1} - \frac{2}{n^2+n}$$

which closely approaches Graham's bound for large n.



$L = (R_0, U_1, U_2, U_3, S_1, S_2,$

$T_0, T_1, T_2, T_3, R_1, R_2)$

Fig. 6. An example of degradation with increasing n, precedence preserved.

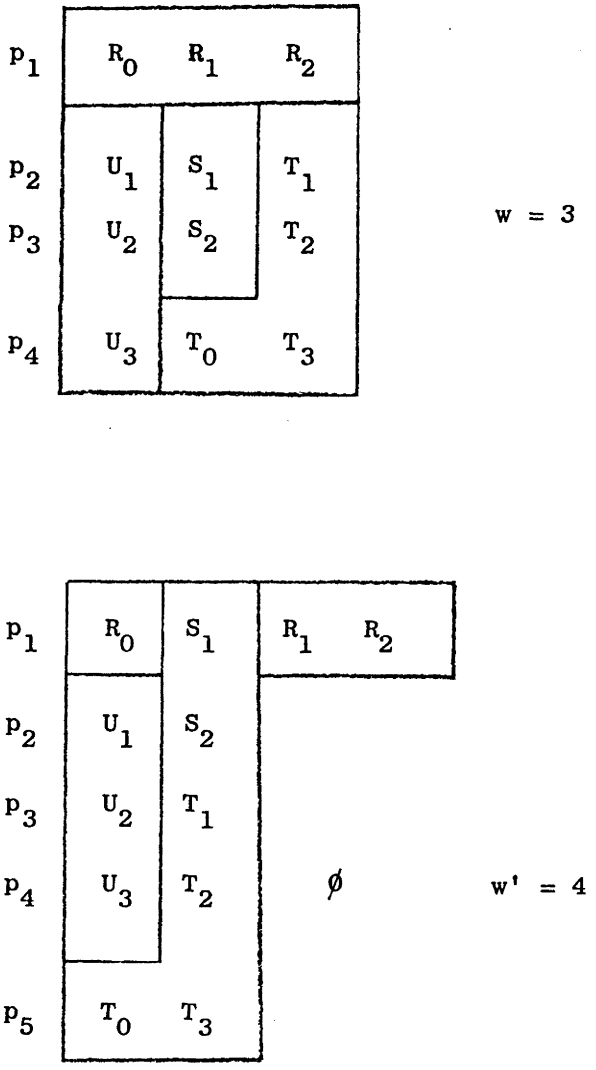
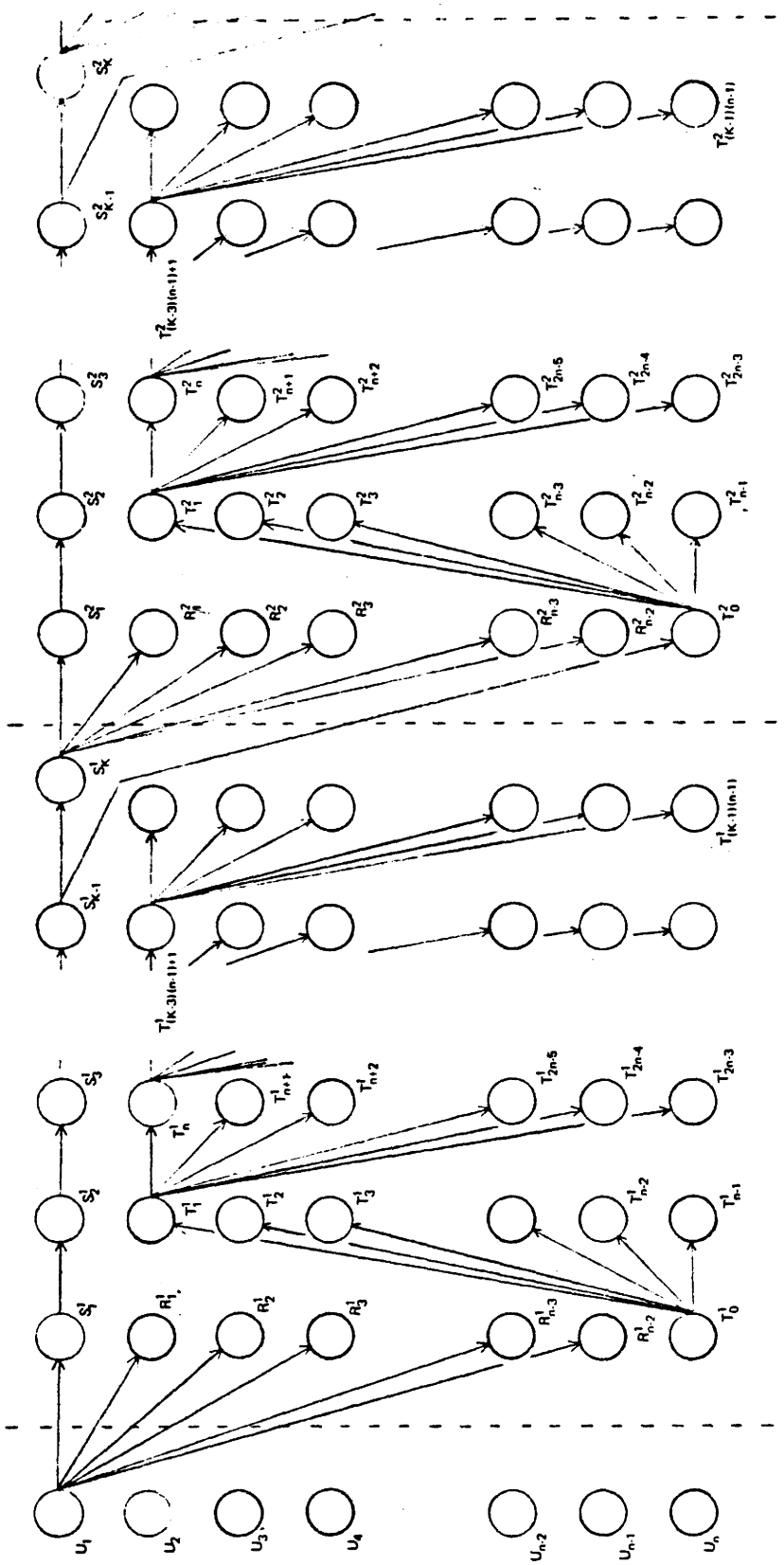


Fig. 7. Gantt charts for the system of Fig. 6.



$L = (U_1, U_2, U_3, \ldots, U_n,$

$R_1^1, R_2^1, \ldots, R_{n-2}^1, T_0^1, T_1^1, T_2^1, \ldots, T_{(K-1)(n-1)}^1), S_1^1, S_2^1, \ldots, S_K^1,$

$R_1^2, R_2^2, \ldots, R_{n-2}^2, T_0^2, T_1^2, T_2^2, \ldots, R_{(K-1)(n-1)}^2), S_1^2, S_2^2, \ldots, S_K^2, \ldots)$

Fig. 8. A system which degrades with increasing n, precedence preserving list.

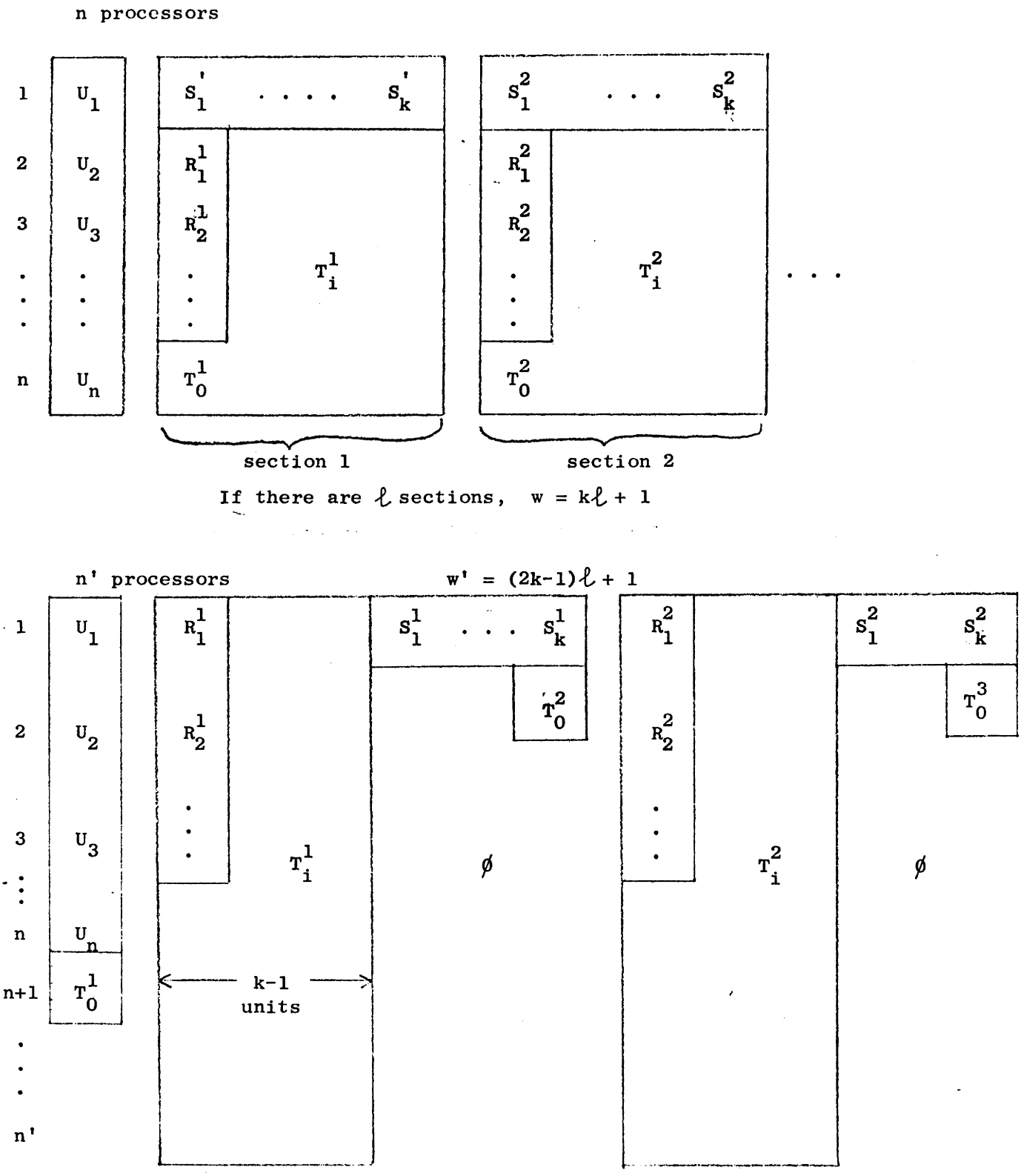


Fig. 9. Gantt charts for the system of Fig. 8.

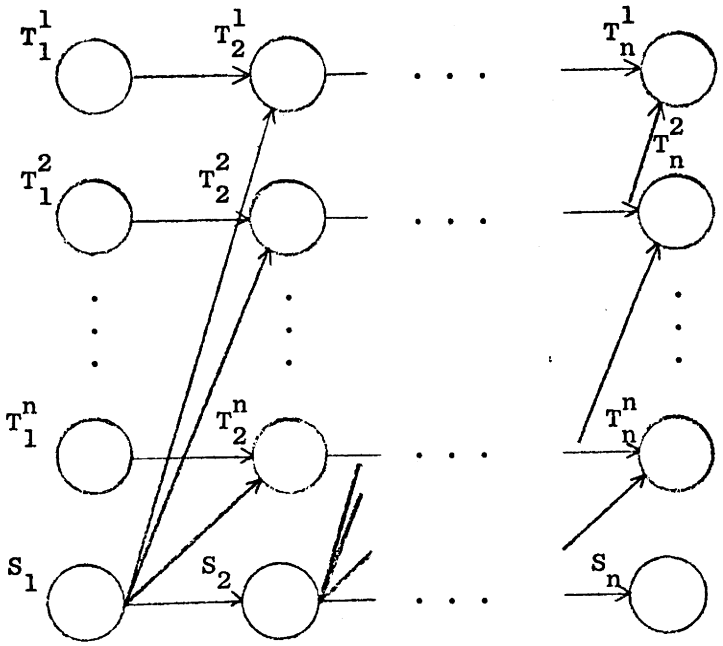
Longest Path Scheduling

One commonly used heuristic constraint is that of "longest path" scheduling. That is, at any time, give preference to the tasks that are farthest from a terminal node in the constraint graph.

The schedules so obtained may still be far from optimal, as the example of Figure 10 (due to Graham [6]) shows. Both L and L' are "longest path" schedules. Execution of schedule L with n processors gives w=n+1. However, execution with schedule L' (again n processors) gives w'=2n. The degradation is $w'/w = 2 - 2/(n+1)$.

Nor does this heuristic protect against degradation with increasing n. Consider the system shown in Figure 11. Gantt charts for n=7 and n'=8 processors are shown in Figure 12. Under longest path scheduling we find a degradation of 1 unit when n is increased.

The above examples indicate that there are systems for which longest path scheduling is always non-optimal. Indeed, such a system is shown in Figure 13. All optimal schedules with 3 processors must include T5 as one of the first 3 tasks.



$$L' = (T_1^1, T_1^2, \ldots, T_1^n, S_1, T_2^1, \ldots, T_n^n, S_n)$$

$$L = (S_1, T_1^1, T_1^2, \ldots, T_1^n, S_2, \ldots, T_n^n)$$

Fig. 10. An example of non-optimal Longest-path scheduling.

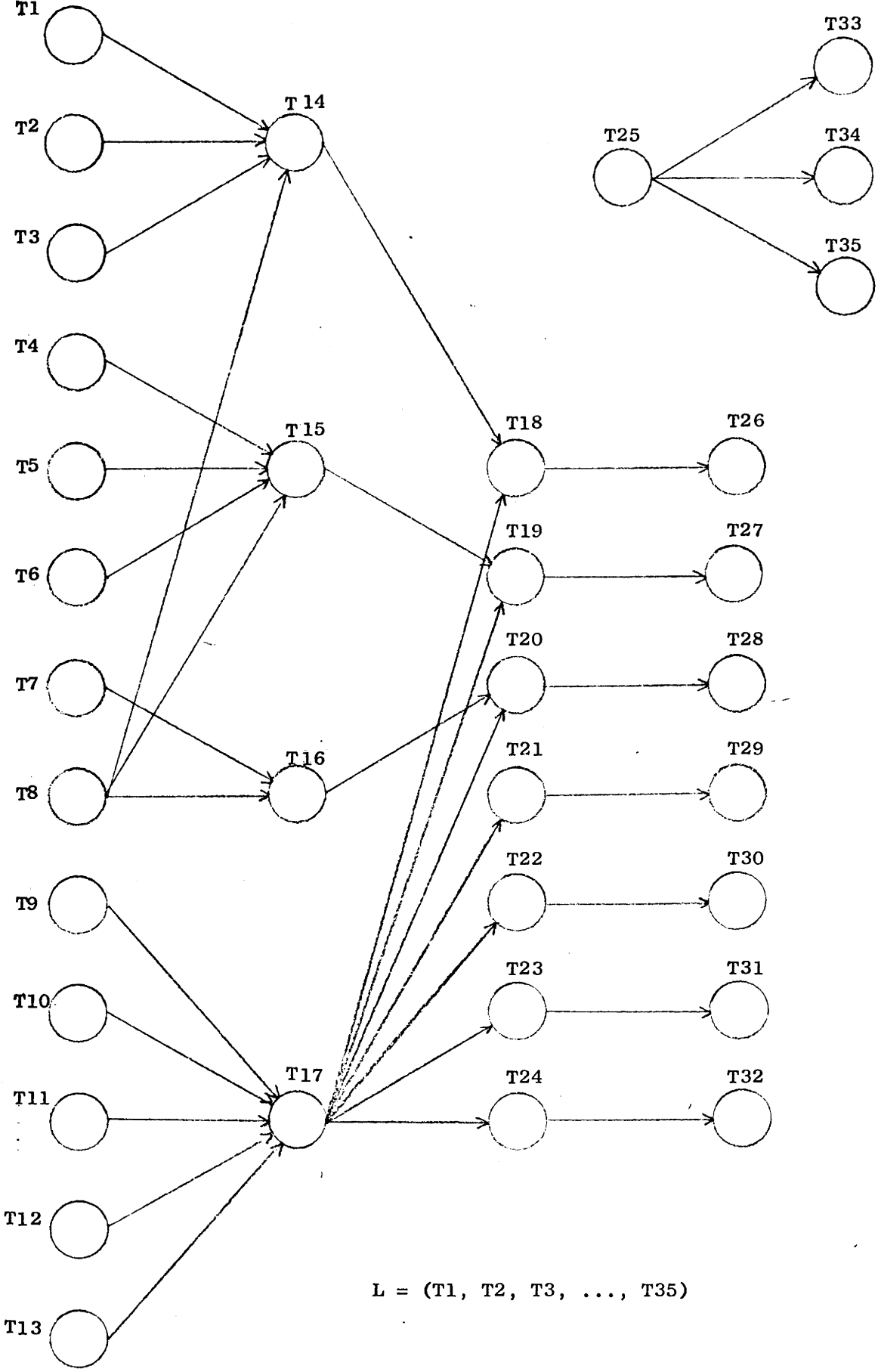


Fig. 11. A system which degrades with increasing n, Longest-path scheduling.

n = 7 processors

1	T1	T8	T14	T18	T26
2	T2	T9	T15	T19	T27
3	T3	T10	T16	T20	T28
4	T4	T11	T17	T21	T29
5	T5	T12	(T33)	T22	T30
6	T6	T13	(T34)	T23	T31
7	T7	(T25)	(T35)	T24	T32

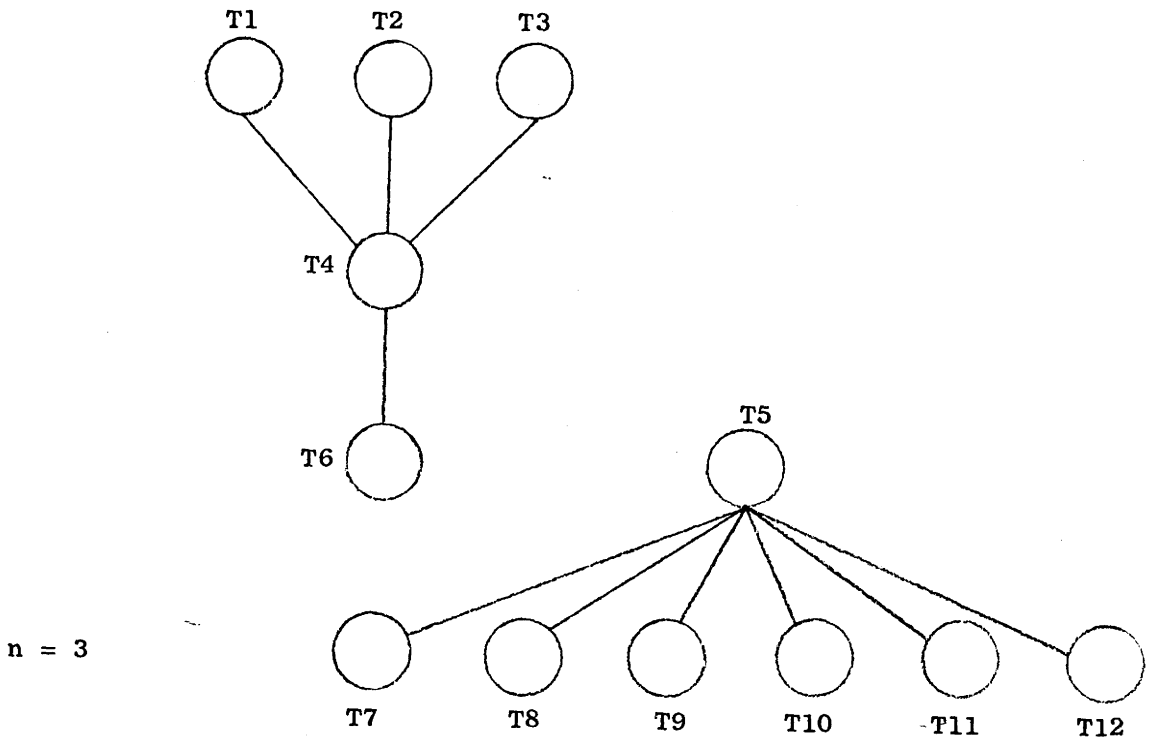
w = 5

n' = 8 processors

1	T1	T9	T17	T18	T26	(T35)
2	T2	T10	(T25)	T19	T27	ø
3	T3	T11	ø	T20	T28	ø
4	T4	T12	ø	T21	T29	ø
5	T5	T13	ø	T22	T30	ø
6	T6	T14	ø	T23	T31	ø
7	T7	T15	ø	T24	T32	ø
8	T8	T16	ø	(T33)	(T34)	ø

w' = 6

Fig. 12. Gantt charts for the system of Fig. 11.



L' = (T1, T2, T3, ..., T12)

L = (T5, T1, ..., T4, T6, ..., T12)

Fig. 13. A system with no optimal Longest-path scheduling.

Conclusions

In this paper we have examined a number of scheduling algorithms for timing anomalies. The examples presented have shown that even heuristics generally thought to be "good", can be highly non-optimal in certain cases.

REFERENCES

[1] Bowdon, E.K., Sr., "Priority Assignment in a Network of Computers," IEEE Trans. on Computers, Vol. C-18, No. 11, Nov. 1969, pp. 1021-1026.

[2] Coffman, E.G., Jr., and R.L. Graham, "Optimal Scheduling for Two-Processor Systems," Acta Informatica 1972.

[3] Fujii, M., T. Kasami and K. Ninomiya, "Optimal Sequencing of Two Equivalent Processors," SIAM J. Appl. Math., Vol. 17, No. 4, July 1969, pp. 784-789.

[4] Graham, R.L., "Bounds for Certain Multiprocessing Anomalies," Bell Syst. Tech J., Vol. 45, No. 9, Nov. 1966, pp. 1563-1581.

[5] Graham, R.L., "Bounds on Multiprocessing Anomalies," SIAM J. Appl. Math., Vol. 17, No. 2, March 1969, pp. 416-429.

[6] Graham, R.L., "Bounds on Multiprocessing Anomalies and Packing Algorithms," Proceedings SJCC, Vol. 40, 1972, pp. 205-217.

[7] Hu, T.C., "Parallel Sequencing and Assembly Line Problems," Operations Research, Vol. 9, No. 6, Nov. 1961, pp. 841-848.

[8] Manacher, G.K., "Production and Stabilization of Real-Time Task Schedules," J. ACM, Vol. 14, No. 3, July 1967, pp. 439-465.

[9] Ramamoorthy, C.V., K.M. Chandy, and M.J. Gonzalez, Jr., "Optimal Scheduling Strategies in Multiprocessor Systems," IEEE Trans. on Computers, Vol. C-21, No. 2, Feb. 1972, pp. 137-146.