Notes on Heuristics

Winter 1978

V. Chvátal

1. Scheduling Independent Tasks

Let us consider a set of identical processors $P_1, P_2, \ldots, P_m$ and a set of tasks $T_1, T_2, \ldots, T_n$. Each task T_i has an execution time t_i and requires only one processor. The processors operate in parallel and none of them can execute more than one task at a time; once it has begun executing T_i, it continues executing it until its completion time t_i time units later.

For example, if $m=3$, $n=11$ and $(t_1, t_2, \ldots, t_{11})$ reads

$$2,4,3,4,4,5,2,1,4,3,2$$

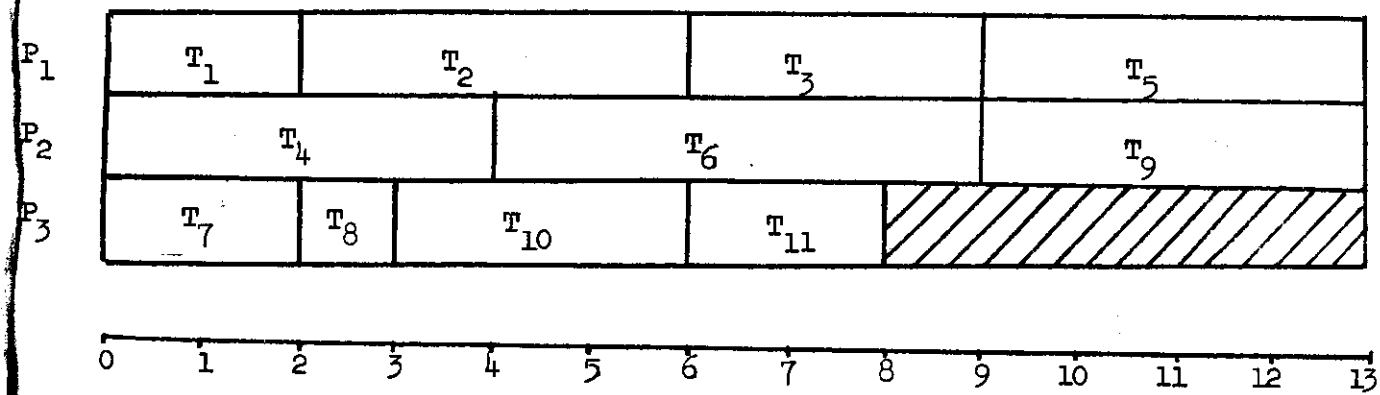
then a possible schedule might go as follows:

P_1 processes T_1, T_2, T_3, T_5

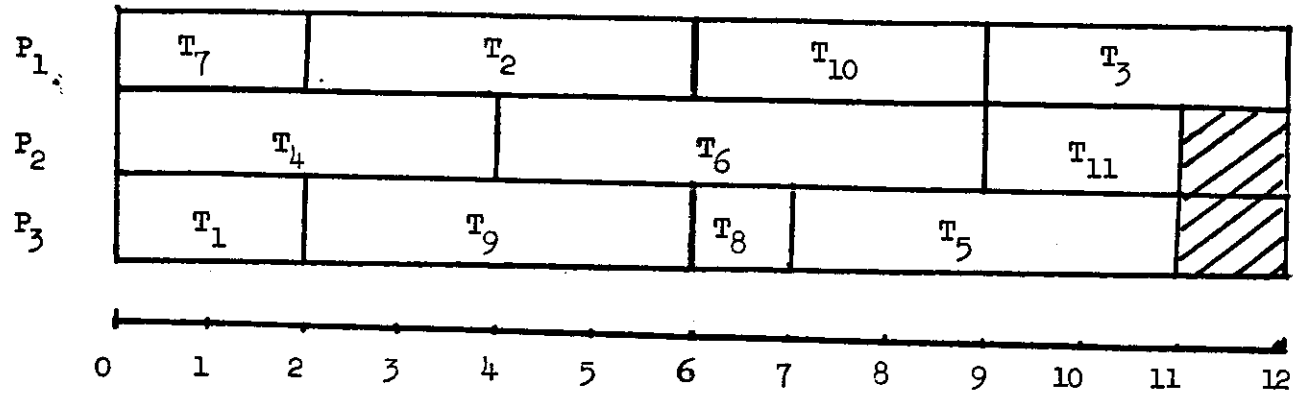
P_2 processes T_4, T_6, T_9

P_3 processes T_7, T_8, T_{10}, T_{11}

This schedule is represented by the diagram shown below.

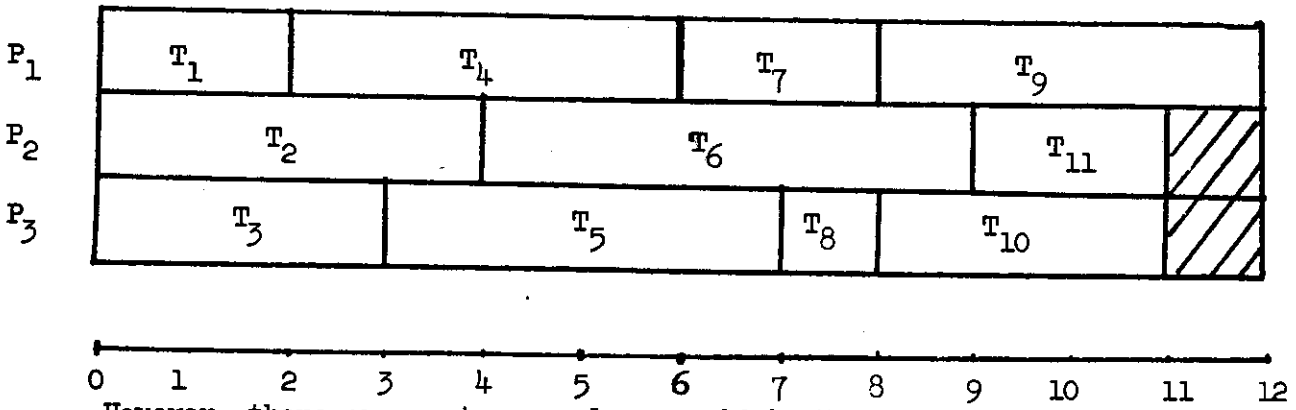


One of the possibly desirable goals is to get all the work done as soon as possible. In this sense, the schedule shown above is inferior to that shown below.

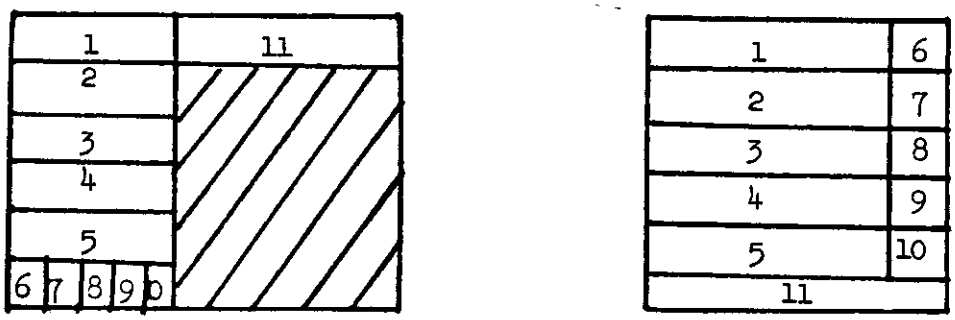


Even when there are only two processors, finding an optimal schedule may be a very difficult problem. Hence it is important to have efficient algorithms which approximate optimal solutions with reasonable accuracy.

The most straightforward of these heuristics is the LIST heuristic: we consider the tasks in the order in which they appear on our list $T_1, \ldots, T_n$ and assign them the processors in such a way that, as soon as a processor becomes idle, it is assigned the next available task. When applied to our example, LIST yields the (optimal) schedule shown below.



However, there are nasty examples on which LIST does not perform that well. The following figure shows the LIST schedule and the optimal schedule for 5,5,5,5,5,1,1,1,1,1,6.



More generally, there are examples with $n=2m-1$ where the finishing time L of LIST equals $2m-1$ but the optimal finishing time OPT is only m. Nevertheless, we shall prove that the ratio L/OPT cannot get any worse than that.

THEOREM 1.1. $\quad L \leq (2 - \frac{1}{m})\ OPT$

The theorem follows immediately from the following lemma combined with the observation that $OPT \geq t$.

LEMMA 1.2. If t is the execution time of a task which finishes last in LIST schedule then

$$L \leq OPT\ (1 + \frac{m-1}{m} \cdot \frac{t}{OPT}).$$

PROOF.

Since no processor is idle before time L-t, we have

$$(\sum_{i=1}^{n} t_i) - t \geq m(L - t).$$

On the other hand,

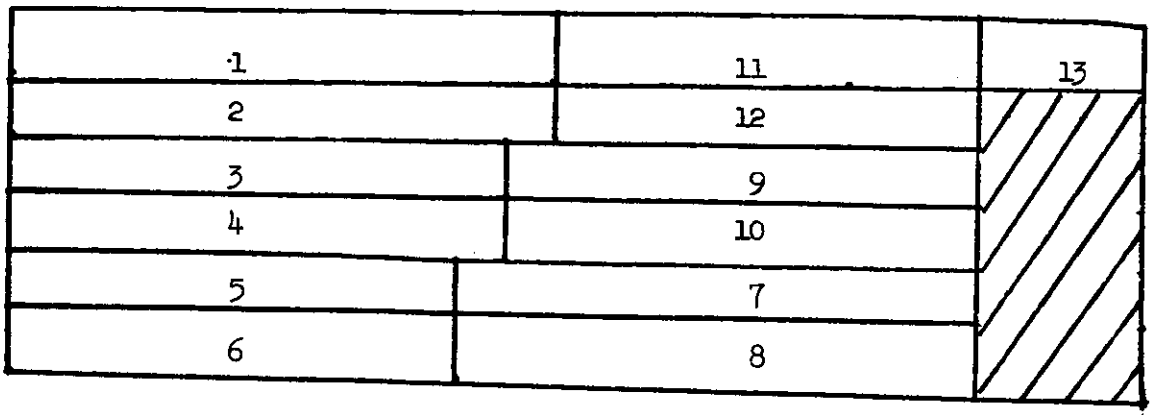
$$OPT \geq \frac{1}{m} \sum_{i=1}^{n} t_i$$

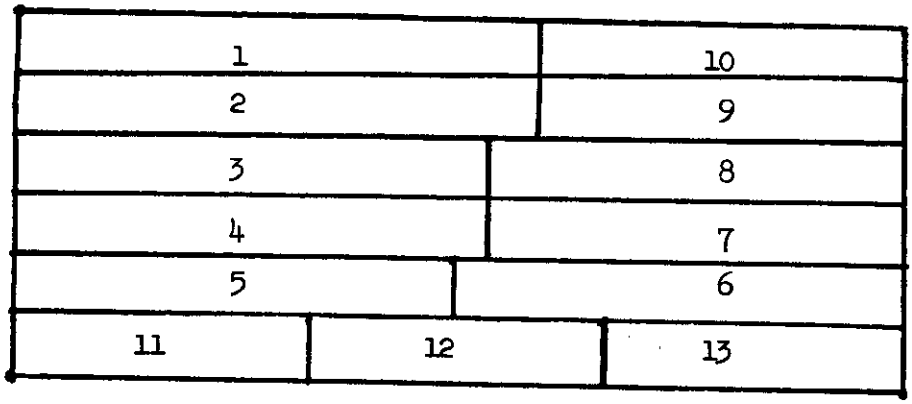
Combining these two inequalities we obtain the desired result.

A slightly more sophisticate heuristic is LIST DECREASING: first order the tasks so that $t_1 \geq t_2 \geq \ldots \geq t_n$ and then apply LIST. An example where LIST DECREASING performs relatively poorly is given by m=6, n=13 and execution times

11,11,10,10,9,9,8,8,7,7,6,6,6.

The LIST DECREASING heuristic delivers a schedule with finishing time LD=23 whereas the optimal finishing time OPT is only 18.





More generally, there are examples with $n=2m+1$, $LD=4m-1$ and $OPT=3m$ for all m. However, the ratio LD/OPT cannot get any worse.

THEOREM 1.3. $LD \leq OPT\ (\frac{4}{3} - \frac{1}{3m})$.

Before proving this theorem, let us establish a simple lemma.

LEMMA 1.4. If $t_1 \geq t_2 \geq \cdots \geq t_n > OPT/3$ then $LD = OPT$.

PROOF. In the optimal schedule, each processor executes at most two tasks. To simplify the formalism, we shall introduce $2m-n$ dummy tasks with execution times zero. Then we may claim that each processor P_i executes precisely two tasks, with execution times a_i and b_i. Without loss of generality we may assume that

$$a_1 \geq a_2 \geq a_3 \geq \cdots \geq a_m$$

and that

$$a_1 \geq b_1,\ a_2 \geq b_2,\ \cdots,\ a_m \geq b_m.$$

Clearly, these inequalities imply $a_1=t_1$. If $a_i=t_i$ for all $i=1,2,\ldots,k-1$ but $a_k<t_k$ then necessarily $t_k=b_i$ for some $i<k$. Interchanging b_i and a_k we obtain an optimal schedule again since $a_i+a_k<a_i+b_i\leq OPT$ and, of course, $b_i+b_k\leq a_i+a_k$. After at most m interchanges of this kind, we obtain an optimal schedule with

$$a_1 = t_1,\ a_2 = t_2,\ \ldots\ ,\ a_m = t_m.$$

Now consider an arbitrary subscript k such that $1 \leq k \leq m$. Since t_{2m+1-k} is the k-th smallest of the m numbers $b_1, b_2, \ldots, b_m$, there must exist a subscript i such that $1 \leq i \leq k$ and $t_{2m+1-k} \leq b_i$. Hence

$$t_k + t_{2m+1-k} \leq a_k + b_i \leq a_i + b_i \leq OPT. \qquad (*)$$

On the other hand, note that

$$t_k + t_{2m+1-k} \geq 2t_n > OPT - t_n$$

and so LIST DECREASING lets each P_k handle T_k and T_{2m+1-k}. But then (*) yields the desired result.

PROOF OF THEOREM 1.3. Consider a counterexample with n as small as possible. Still assuming $t_1 \geq t_2 \geq \ldots \geq t_n$, observe that T_n finishes last in the LIST DECREASING schedule. (Otherwise deletion of T_n from our list would leave LD unchanged and produce a counterexample with a smaller value of n.) Now Lemma 1.2 with $t=t_n$, and the assumption that we are working with a counterexample, combine into

$$\frac{4}{3} - \frac{1}{3m} < \frac{LD}{OPT} \leq 1 + \frac{m-1}{m} \cdot \frac{t_n}{OPT} .$$

Hence $t_n > OPT/3$. But then Lemma 1.4 implies $LD = OPT$ which is a contradiction.