Discrete Applied Mathematics 10 (1985) 155–164
North-Holland

ANALYSIS OF A LINEAR PROGRAMMING HEURISTIC FOR SCHEDULING UNRELATED PARALLEL MACHINES

C.N. POTTS
University of Keele, Keele, Staffordshire ST5 5BG, U.K.

Received 15 January 1984

Each of n jobs is to be processed without interruption on one of m unrelated parallel machines. The objective is to minimize the maximum completion time. A heuristic method is presented, the first stage of which uses linear programming to form a partial schedule leaving at most $m-1$ jobs unscheduled: the second stage schedules these $m-1$ jobs using an enumerative method. For $m \geq 3$, it is shown that the heuristic has a (best possible) worst-case performance ratio of 2 and has a computational requirement which is polynomial in n although it is exponential in m. For $m=2$, it is shown that the heuristic has a (best possible) worst-case performance ratio of $(1+\sqrt{5})/2$ and requires linear time. A modified version of the heuristic is presented for $m=2$ which is shown to have a (best possible) worst-case performance ratio of 3/2 while still requiring linear time.

1. Introduction

The problem that is considered in this paper of scheduling jobs on unrelated parallel machines may be stated as follows. Each of n jobs (numbered $1, \ldots, n$) is to be processed without interruption on one of m machines (numbered $1, \ldots, m$). At any time, each machine can process at most one job. Job j ($j=1, \ldots, n$) becomes available for processing at time zero and requires a positive *processing time* p_{ij} if it is scheduled on machine i ($i=1, \ldots, m$). The objective is to schedule the jobs so that the maximum completion time is minimized.

When $p_{ij}=p_j$ ($i=1, \ldots, m; j=1, \ldots, n$), where the processing requirement p_j of job j is the same for each machine, then the machines are *identical*. When $p_{ij}=p_j/q_i$ ($i=1, \ldots, m; j=1, \ldots, n$), where p_j is the processing requirement of job j and q_i is the speed of machine i, then the machines are *uniform*. Because in our general problem the matrix of processing times has arbitrary positive entries, the machines are *unrelated*. Karp [9] shows that for the case of two identical machines the problem is NP-hard which indicates that the existence of a polynomial bounded algorithm to solve the problem is highly unlikely. Consequently, for parallel machine scheduling problems of this type, most researchers have studied heuristic methods which provide an approximate solution. Suppose that $C^*_{\max}$ denotes the minimum value of the maximum completion time, while $C^H_{\max}$ denotes the value of the maximum completion time when the jobs are scheduled using a certain heuristic H. If, whatever the problem data, $C^H_{\max}/C^*_{\max} \leq \varrho$ for a specified constant ϱ, where ϱ is as small as

possible, then ϱ is called the *worst-case performance ratio* of H. A survey and discussion of the worst-case analysis of heuristics are given by Fisher [3] and Garey et al. [4].

Horowitz and Sahni [7] present algorithms A_ε that require $O(n(n^2/\varepsilon)^{m-1})$ time for which $C_{\max}^{A_\varepsilon}/C_{\max}^* \leq 1+\varepsilon$. Several heuristics are analyzed by Ibarra and Kim [8], the most promising of which is the earliest completion time heuristic ECT which requires $O(mn^2)$ time. In heuristic ECT, at each stage an unscheduled job is chosen so that when it is scheduled on the machine which can complete it earliest, its completion time is as small as possible. This chosen job is then scheduled on that machine that can complete it earliest and the process is repeated until a complete schedule is obtained. Ibarra and Kim show that $C_{\max}^{\mathrm{ECT}}/C_{\max}^* \leq m$ and construct an example for which $C_{\max}^{\mathrm{ECT}}/C_{\max}^*$ can be arbitrarily close to 2. Davis and Jaffe [2] limit the worst-case performance ratio of ECT by constructing an example for which $C_{\max}^{\mathrm{ECT}}/C_{\max}^* = 1+\log_2 m$ when m is a power of 2. For the special case $m=2$, Ibarra and Kim present a heuristic that has a worst-case performance ratio of $(1+\sqrt{5})/2$ and which requires $O(n \log n)$ time. Davis and Jaffe propose several heuristics each of which is shown to have a worst-case performance ratio of $O(\sqrt{m})$. Their strongest result shows that the worst-case performance ratio of one of the heuristics is bounded above by $3\sqrt{m}/2+2+1/(2\sqrt{m})$.

In Section 2 of this paper we present a heuristic, the first stage of which uses linear programming to form a partial schedule leaving at most $m-1$ jobs unscheduled; the second stage schedules these $m-1$ jobs using an enumerative method. It is shown that this heuristic has a worst-case performance ratio of 2 when $m \geq 3$. Section 3 provides an analysis of the heuristic for the case $m=2$ and shows that it has a worst-case performance ratio of $(1+\sqrt{5})/2$ which, through a minor modification to the heuristic, can be reduced to 3/2. Some concluding remarks are given in Section 4.

2. The linear programming heuristic

It is clear that if a given schedule is modified by reordering the jobs on any of the machines, then the maximum completion time is unaltered. Thus, our scheduling problem reduces to one of assigning jobs to machines.

A formulation of the problem is presented which uses zero-one variables x_{ij} $(i=1,\ldots,m;\ j=1,\ldots,n)$, where

$$x_{ij} = \begin{cases} 1 & \text{if job } j \text{ is assigned to machine } i, \\ 0 & \text{otherwise.} \end{cases}$$

and the variable $C_{\max}$ which represents the maximum completion time. We refer to the x_{ij} as *assignment variables*. The problem can be written as

$$\text{minimize} \quad C_{\max},$$

$$\text{subject to} \quad \sum_{j=1}^{n} p_{ij} x_{ij} \leq C_{\max} \quad (i = 1, \ldots, m), \tag{1}$$

$$\sum_{i=1}^{m} x_{ij} = 1 \quad (j = 1, \ldots, n), \tag{2}$$

$$x_{ij} \in \{0, 1\} \quad (i = 1, \ldots, m; j = 1, \ldots, n). \tag{3}$$

Constraints (1) ensure that $C_{\max}$ is at least as large as the total processing time on any machine, while constraints (2) and (3) ensure that each job is processed on exactly one machine. Let $C^*_{\max}$ denote the value of an optimal solution.

Consider the *linear programming relaxation* in which constraints (3) are removed and replaced by $x_{ij} \geq 0$ $(i = 1, \ldots, m; j = 1, \ldots, n)$. (The restriction $x_{ij} \leq 1$ $(i = 1, \ldots, m; j = 1, \ldots, n)$ follows from (2).) The effect of relaxing the integrality conditions in this way is to allow the processing of a job to be shared amongst the machines. In contrast to the corresponding preemptive scheduling problem, however, several machines may simultaneously process the same job. A *partial schedule* is obtained from the solution of the linear programming relaxation by assigning job j to machine i whenever $x_{ij} = 1$ $(i = 1, \ldots, m; j = 1, \ldots, n)$. Any jobs not appearing in this partial schedule are called *fractional jobs* due to the non-integrality of the corresponding assignment variables. If we are fortunate, the solution of the linear programming relaxation and of the original problem are identical in which case the partial schedule is a complete optimal schedule and there are no fractional jobs. The following analysis gives, however, an upper bound on the number of fractional jobs.

The linear programming problem has $m + n$ constraints in addition to the non-negativity conditions. Therefore, there exists an optimal basic solution having $m + n$ basic variables which may take positive values while the other non-basic variables take the value zero. The simplex method always generates a basic solution of this form. However, should an alternative linear programming algorithm be applied to yield an optimal solution in which more than $m + n$ variables take positive values, then a polynomial time procedure (for example see Hadley [6]) can be applied to transform the solution into an optimal basic solution. In an optimal basic solution, we have $C_{\max} > 0$ which implies that $C_{\max}$ is a basic variable. It follows that at most $m + n - 1$ of the assignment variables are basic. (It can be shown that each constraint (1) is satisfied as an equality which implies that exactly $m + n - 1$ of the assignment variables are basic.) Since no pair of constraints (2) contains the same variable, it is possible to select a set B of n basic assignment variables corresponding to these constraints such that each constraint (2) contains exactly one basic variable from this set. There are at most a further $m - 1$ basic assignment variables. Let B' denote the set of these further basic assignment variables. When $n \geq m - 1$, at most $m - 1$ of the constraints (2) contain a basic variable from B' in addition to a basic variable from B, while at least $n - m + 1$ of them contain one basic variable from B and no basic variable from B'. In those constraints containing exactly one basic variable, the basic variable takes the value one since the other non-basic variables each take

the value zero. Thus, for $n \geq m-1$, at least $n-m+1$ variables take the value one and consequently at least $n-m+1$ jobs appear in the partial schedule, leaving at most $m-1$ jobs unscheduled.

A complete description of the algorithm is presented next. The linear programming relaxation is solved first and, if necessary, the solution is then transformed into an optimal basic solution. A partial schedule is formed from those assignment variables which take the value one. (For the case $n \leq m-1$, the partial schedule may be empty.) A complete schedule is produced by appending the fractional jobs to the partial schedule. All possible assignments of fractional jobs to machines are considered and one which gives a complete schedule having the smallest maximum completion time is chosen. This heuristic which consists of linear programming and enumeration is referred to as heuristic LPE.

We now discuss the computational complexity of heuristic LPE. The linear programming problem involving $mn+1$ variables and $m+n$ constraints can be polynomially solved by Khachian's algorithm [10]. Any transformation of the solution to a basic solution also requires polynomial time. Since there are at most $m-1$ fractional jobs which need to be assigned to machines using complete enumeration, a maximum of m^{m-1} schedules are generated and compared in the second stage of the heuristic. Thus, for fixed m the number of computational steps required by the heuristic is polynomial, although for arbitrary m an exponential number of steps are required.

The worst-case performance ratio of our heuristic is derived next. In this section the case $m \geq 3$ is considered, while in the next section we analyze the case $m=2$.

Theorem 1. $C_{\max}^{\mathrm{LPE}}/C_{\max}^{*} \leq 2$ *and, for* $m \geq 3$, *this bound is the best possible.*

Proof. Let $C_{\max}^{\mathrm{LP}}$ denote the value of an optimal solution of the linear programming relaxation and let $C_{\max}^{\mathrm{E}}$ denote the maximum completion time of an optimal schedule of the fractional jobs when considered in isolation. Since the linear programming relaxation provides a lower bound, we have

$$C_{\max}^{*} \geq C_{\max}^{\mathrm{LP}}. \tag{4}$$

Also, an optimal schedule of the fractional jobs provides a lower bound on the maximum completion time of a full schedule. Therefore,

$$C_{\max}^{*} \geq C_{\max}^{\mathrm{E}}. \tag{5}$$

One possible schedule that could be generated by heuristic LPE is to append the optimal schedule of the fractional jobs considered in isolation to the partial schedule obtained from the solution of the linear programming relaxation. The schedule actually selected by heuristic LPE is at least as good since the fractional jobs are assigned to give a maximum completion time which is as small as possible. Thus,

$$C_{\max}^{\mathrm{LPE}} \leq C_{\max}^{\mathrm{LP}} + C_{\max}^{\mathrm{E}}. \tag{6}$$

It follows from (4), (5) and (6) that $C_{\max}^{\mathrm{LPE}}/C_{\max}^{*} \leq 2$.

To show that this bound is the best possible we consider an example in which there are m uniform machines. There are m jobs and $p_{ij} = p_j/q_i$ $(i, j = 1, \ldots, m)$, where $q_1 = m-1$, $q_i = 1$ for $i = 2, \ldots, m$, $p_1 = (m-1)p$ and $p_j = p$ for $j = 2, \ldots, m$ for any positive p. An optimal schedule is obtained by assigning job j to machine j for $j = 1, \ldots, m$ giving $C_{\max}^{*} = p$. A solution (not unique) of the linear programming relaxation is $x_{11} = 0$, $x_{1j} = 1$ $(j = 2, \ldots, m)$, $x_{i1} = 1/(m-1)$ $(i = 2, \ldots, m)$ and $x_{ij} = 0$ $(i, j = 2, \ldots, m)$. The partial schedule assigns jobs $2, \ldots, m$ to machine 1 and job 1 is the fractional job. Enumeration shows that job 1 is scheduled on machine 1 to give $C_{\max}^{\mathrm{LPE}} = 2p$. Therefore, $C_{\max}^{\mathrm{LPE}}/C_{\max}^{*} = 2$ as required. □

It should be noted that although the bound on $C_{\max}^{\mathrm{LPE}}/C_{\max}^{*}$ given in Theorem 1 is valid for $m = 2$, it is not in this case the best possible. For the example presented in the proof, when $m = 2$ the enumeration procedure is not required since there are no fractional jobs.

For practical purposes the linear programming problem would be solved by the simplex method rather than Khachian's algorithm. Because of its simple structure, the number of simplex iterations required to solve the problem is unlikely to be large. In the second stage of the heuristic, the complete enumeration procedure can be replaced by a branch and bound algorithm. Such use of a bounding procedure to limit the search reduces computational requirements for many problems. A branch and bound algorithm has the further advantage that, if desired, computation can be terminated before optimality is reached. Since an optimal solution is often generated at an early stage of a branch and bound algorithm while the remaining computation verifies optimality, early termination of the algorithm does not necessarily detract from its performance although the guarantee of Theorem 1 becomes invalid.

An alternative heuristic LPH can be designed by scheduling the $m-1$ or fewer fractional jobs using some heuristic H instead of by complete enumeration. Then, using the same arguments as in the proof of Theorem 1, we deduce that $C_{\max}^{\mathrm{LPH}}/C_{\max}^{*} \leq 1 + C_{\max}^{\mathrm{H}}/C_{\max}^{*}$. Ideally, heuristic H should be capable of scheduling $m-1$ jobs in polynomial time and, for $(m-1)$-job scheduling problems, have a low worst-case performance ratio. Unfortunately, in terms of worst-case performance ratios, there is no apparent advantage in incorporating the heuristics of Ibarra and Kim [8] or of Davis and Jaffe [2] into LPH rather than using them alone. Further research is required to find a suitable heuristic H for use in LPH.

In spite of its extra computational requirements, the worst-case performance ratio of heuristic LPE, which is independent of m, makes it, for many applications, more attractive than the heuristics of Ibarra and Kim [8] and of Davis and Jaffe [2]. The exponential computational requirement of $O(n(n^2/\varepsilon)^{m-1})$ time for the approximation scheme of Horowitz and Sahni [7] deems it to be more restrictive for larger values of n than heuristic LPE. We conclude that there are situations in which heuristic LPE would be used in preference to these other heuristic methods.

3. Analysis of heuristic LPE for $m=2$

All the results and the discussion in this section are confined to the problem of scheduling two unrelated machines. We commence the analysis of heuristic LPE by describing the efficient method of Gonzalez et al. [5] for solving the linear programming relaxation. The following notation is adopted. For any set of jobs S, let $p_i(S)$ denote the total processing time on machine i ($i=1,2$) of the jobs of S.

The first step in solving the linear programming problem is to compute the *ratio* p_{1j}/p_{2j} for each job j. It is assumed that all ratios are distinct; if for job j_1 and job j_2, where $j_1<j_2$, a tie occurs, then the ratio for job j_1 is considered to be smaller. Then, job k is found together with the corresponding sets $S_{1k}=\{j \mid p_{1j}/p_{2j}< p_{1k}/p_{2k}\}$ and $S_{2k}=\{j \mid p_{1j}/p_{2j}>p_{1k}/p_{2k}\}$ such that $p_1(S_{1k})+p_{1k}\geq p_2(S_{2k})$ and $p_1(S_{1k})< p_2(S_{2k})+p_{2k}$. Job k is called the *dividing job*. Gonzalez et al. [5] show that the linear programming problem is solved by setting

$$C_{\max}=(p_{1k}p_2(S_{2k})+p_{2k}p_1(S_{1k})+p_{1k}p_{2k})/(p_{1k}+p_{2k}),$$
$$x_{1j}=1,\ x_{2j}=0 \quad \text{for } j\in S_{1k},$$
$$x_{1j}=0,\ x_{2j}=1 \quad \text{for } j\in S_{2k},$$
$$x_{1k}=(p_2(S_{2k})+p_{2k}-p_1(S_{1k}))/(p_{1k}+p_{2k}),$$
$$x_{2k}=(p_1(S_{1k})+p_{1k}-p_2(S_{2k}))/(p_{1k}+p_{2k}).$$

Thus, for the special case in which $p_1(S_{1k})+p_{1k}=p_2(S_{2k})$, there is no fractional job and the partial schedule in which the jobs of $S_{1k}\cup\{k\}$ are assigned to machine 1 and the jobs of S_{2k} are assigned to machine 2 is a complete optimal schedule. On the other hand, when $p_1(S_{1k})+p_{1k}>p_2(S_{2k})$, job k is a fractional job and the partial schedule assigns the jobs of S_{ik} to machine i ($i=1,2$): the complete schedule obtained by assigning job k to machine 1 and that obtained by assigning it to machine 2 are compared thereby giving a complete schedule with maximum completion time

$$C_{\max}^{\mathrm{LPE}}=\min\{p_1(S_{1k})+p_{1k},\ p_2(S_{2k})+p_{2k}\}.$$

We show next that the dividing job k and consequently the sets S_{1k} and S_{2k} can be found in linear time. All the ratios p_{1j}/p_{2j} can be computed in $O(n)$ time and job l is found together with sets S_{1l} and S_{2l} such that $|S_{1l}|=|S_{2l}|$ or $|S_{1l}|=|S_{2l}|-1$. This can be implemented in $O(n)$ time using a median finding routine [1, 13]. If $p_1(S_{1l})+p_{1l}=p_2(S_{2l})$, then l is the dividing job and we proceed no further. On the other hand, if $p_1(S_{1l})+p_{1l}>p_2(S_{2l})$, the search for the dividing job is restricted to the jobs of $S_{1l}\cup\{l\}$ with the jobs of S_{2l} forming a subset of S_{2k} and if $p_1(S_{1l})+p_{1l}< p_2(S_{2l})$, the search for the dividing job is restricted to the jobs of S_{2l} with the jobs of $S_{1l}\cup\{l\}$ forming a subset of S_{1k}. In either case, the search for the dividing job k is restricted to a subset of jobs which contains one half of the original jobs. Reapplying the procedure requires one half of the computational steps required by the first application and restricts the search to a subset of jobs containing one quarter

of the original jobs. After $O(\log n)$ applications of the procedure that are carried out over sets which contain n, $n/2$, $n/4, \ldots$ jobs, job k is found in $O(n)$ time.

We proceed by deriving the worst-case performance ratio of heuristic LPE.

Theorem 2. *For* $m=2$, $C_{\max}^{\mathrm{LPE}}/C_{\max}^{*} \leq (1+\sqrt{5})/2$ *and this bound is the best possible.*

Proof. As before, let $C_{\max}^{\mathrm{LP}}$ denote the value of the solution of the linear programming relaxation obtained using the procedure described above. The corresponding partial schedule of jobs is completed on machine 1 at time $p_1(S_{1k}) = C_{\max}^{\mathrm{LP}} - p_{1k}x_{1k}$ and is completed on machine 2 at time $p_2(S_{2k}) = C_{\max}^{\mathrm{LP}} - p_{2k}x_{2k}$. Since $x_{1k}+x_{2k}=1$, we have

$$C_{\max}^{\mathrm{LPE}} = C_{\max}^{\mathrm{LP}} + \min\{p_{1k}(1-x_{1k}), p_{2k}x_{1k}\}. \tag{7}$$

Clearly, the total processing time of job k on machine 2 as required in the solution of the linear programming problem cannot exceed $C_{\max}^{\mathrm{LP}}$. Therefore,

$$p_{2k}(1-x_{1k}) \leq C_{\max}^{\mathrm{LP}}.$$

Substituting in (7) yields

$$C_{\max}^{\mathrm{LPE}} \leq C_{\max}^{\mathrm{LP}} + \min\{p_{1k}(1-x_{1k}), x_{1k}C_{\max}^{\mathrm{LP}}/(1-x_{1k})\}.$$

Assume without loss of generality that $p_{1k} \leq p_{2k}$. Then $p_{1k} \leq C_{\max}^{*}$ and, as in Section 2, $C_{\max}^{\mathrm{LP}} \leq C_{\max}^{*}$. Hence,

$$C_{\max}^{\mathrm{LPE}}/C_{\max}^{*} \leq 1 + \min\{1-x_{1k}, x_{1k}/(1-x_{1k})\}.$$

For $0 \leq x_{1k} \leq 1$, $1-x_{1k}$ is a decreasing function and $x_{1k}/(1-x_{1k})$ is an increasing function. Thus the minimization term takes its maximum value when $1-x_{1k} = x_{1k}/(1-x_{1k})$. This occurs when $x_{1k} = (3-\sqrt{5})/2$ to give

$$C_{\max}^{\mathrm{LPE}}/C_{\max}^{*} \leq (1+\sqrt{5})/2,$$

as required.

We again consider an example with uniform machines to show that this bound is the best possible. There are 2 jobs with $p_{11} = (-1+\sqrt{5})p/2$, $p_{12} = p$, $p_{21} = p$ and $p_{22} = (1+\sqrt{5})p/2$ for any positive p. The optimal schedule assigns job 1 to machine 2 and assigns job 2 to machine 1 to give $C_{\max}^{*} = p$. The heuristic chooses $k=2$ with $S_{1k} = \{1\}$ and $S_{2k} = \emptyset$. Thus, $C_{\max}^{\mathrm{LPE}} = (1+\sqrt{5})p/2$ to give $C_{\max}^{\mathrm{LPE}}/C_{\max}^{*} = (1+\sqrt{5})/2$. □

For two machines our heuristic is similar to the one of Ibarra and Kim [8] and has the same worst-case performance ratio. However, the Ibarra and Kim heuristic does require the jobs to be ordered according to the ratios p_{1j}/p_{2j}. Thus, its computational complexity is $O(n \log n)$ which makes it less attractive than our method which can be implemented in $O(n)$ time.

An obvious way in which the worst-case performance ratio might be improved is to first apply the heuristic as described above and, assuming that job k is a fractional job, to then reapply it to the modified problem which has the added constraint that

job k is assigned to the machine on which it has the smaller processing time. Assume that the machines are renumbered, if necessary, so that $p_{1k} \leq p_{2k}$. To force job k to be assigned to machine 1, we set $p_{2k} = M$, where M is sufficiently large to ensure that the ratio for job k is smaller than all other ratios. Then, heuristic LPE is re-applied to yield a dividing job k' with corresponding sets $S_{1k'}$ and $S_{2k'}$. The better of the schedules generated by the two applications of heuristic LPE is selected. Although two applications of heuristic LPE are necessary, this modified heuristic, which we denote by LPE′, still requires $O(n)$ time.

The worst-case performance ratio of heuristic LPE′ is derived next.

Theorem 3. $C_{\max}^{\mathrm{LPE}'}/C_{\max}^{*} \leq 3/2$ *and this bound is the best possible.*

Proof. Consider first the case in which there exists an optimal schedule in which job k is assigned to machine 2. Since the schedule generated by heuristic LPE′ is at least as good as the one generated by heuristic LPE, we have, as in the proof of Theorem 2, that

$$C_{\max}^{\mathrm{LPE}'} \leq C_{\max}^{\mathrm{LP}} + \min\{p_{1k}(1-x_{1k}), p_{2k}x_{1k}\}.$$

Recall that the machines are renumbered so that $p_{1k} \leq p_{2k}$. Because there exists an optimal schedule in which job k is assigned to machine 2, it follows that $C_{\max}^{*} \geq p_{2k} \geq p_{1k}$. Also $C_{\max}^{*} \geq C_{\max}^{\mathrm{LP}}$. Therefore,

$$C_{\max}^{\mathrm{LPE}'}/C_{\max}^{*} \leq 1 + \min\{1-x_{1k}, x_{1k}\}.$$

It is clear that the minimization term takes its maximum value when $x_{1k} = 1/2$ to give $C_{\max}^{\mathrm{LPE}'}/C_{\max}^{*} \leq 3/2$ as required for this first case.

We now consider the alternative case in which there exists an optimal schedule in which job k is assigned to machine 1. For the second application of heuristic LPE let job k' be the dividing job and let $C_{\max}^{\mathrm{LP}'}$ denote the value of the solution of the second linear programming problem. For the special case in which $k' = k$, it is clear that the second application of heuristic LPE assigns job k to machine 1 and assigns all other jobs to machine 2 to give a schedule with a maximum completion time of p_{1k}. Since $p_{1k} \leq p_{2k}$, this schedule is optimal. Consequently, we may assume henceforth that $k \neq k'$. It follows from the description of heuristic LPE′ that

$$\begin{aligned} C_{\max}^{\mathrm{LPE}'} \leq \min\{&C_{\max}^{\mathrm{LP}} + p_{1k}(1-x_{1k}),\ C_{\max}^{\mathrm{LP}} + p_{2k}x_{1k}, \\ &C_{\max}^{\mathrm{LP}'} + p_{1k'}(1-x_{1k'}),\ C_{\max}^{\mathrm{LP}'} + p_{2k'}x_{1k'}\}. \end{aligned} \tag{8}$$

We have $C_{\max}^{*} \geq C_{\max}^{\mathrm{LP}}$ and, because in this case forcing job k to be sequenced on machine 1 does not affect the maximum completion time, $C_{\max}^{*} \geq C_{\max}^{\mathrm{LP}'}$. Suppose first that there exists an optimal schedule in which job k' is also assigned to machine 1. Since job k and job k' are both assigned to machine 1 in an optimal schedule, it follows that

$$C_{\max}^{*} \geq p_{1k} + p_{1k'}. \tag{9}$$

We have from (8) that

$$C_{\max}^{\mathrm{LPE}'} \leq \min\{C_{\max}^{*} + p_{1k}(1 - x_{1k}),\ C_{\max}^{*} + p_{1k'}(1 - x_{1k'})\}.$$

The right hand side takes its maximum value when $x_{1k} = x_{1k'} = 0$. Using (9) it follows that

$$C_{\max}^{\mathrm{LPE}'} \leq \min\{C_{\max}^{*} + p_{1k},\ 2C_{\max}^{*} - p_{1k}\}.$$

The minimization takes its maximum value when $p_{1k} = (1/2)C_{\max}^{*}$ to yield $C_{\max}^{\mathrm{LPE}'}/C_{\max}^{*} \leq 3/2$. We finally analyze the case in which there exists an optimal schedule in which job k is assigned to machine 1 and job k' is assigned to machine 2. In the second application of heuristic LPE we observe that $k' \in S_{1k}$ since the second dividing job is always chosen from the set $S_{1k} \cup \{k\}$. Thus, $p_{1k'}/p_{2k'} \leq p_{1k}/p_{2k}$ which implies that $p_{1k'} \leq p_{2k'}$. From (8) we have

$$C_{\max}^{\mathrm{LPE}'} \leq C_{\max}^{\mathrm{LP}'} + \min\{p_{1k'}(1 - x_{1k'}),\ p_{2k'}x_{1k'}\}.$$

Since $C_{\max}^{*} \geq p_{2k'} \geq p_{1k'}$ and $C_{\max}^{*} \geq C_{\max}^{\mathrm{LP}'}$, it follows that

$$C_{\max}^{\mathrm{LPE}'}/C_{\max}^{*} \leq 1 + \min\{1 - x_{1k'}, x_{1k'}\}.$$

The minimization takes its maximum value when $x_{1k'} = 1/2$ to give $C_{\max}^{\mathrm{LPE}'}/C_{\max}^{*} \leq 3/2$. Thus, we have established that $C_{\max}^{\mathrm{LPE}'}/C_{\max}^{*} \leq 3/2$ in all cases.

The following example with three jobs demonstrates that the bound is the best possible. The processing times are $p_{11} = p/2$, $p_{12} = p$, $p_{13} = p/2$, $p_{21} = p$, $p_{22} = p$ and $p_{23} = p/2$ for any positive p. The optimal schedule assigns job 1 and job 3 to machine 1 and assigns job 2 to machine 2 to give $C_{\max}^{*} = p$. The first application of heuristic LPE gives $k = 2$ with $S_{1k} = \{1\}$ and $S_{2k} = \{3\}$ with $C_{\max}^{\mathrm{LPE}} = 3p/2$. After setting $p_{22} = M$, for large M, the second application of heuristic LPE gives $k' = 1$ with $S_{1k'} = \{2\}$ and $S_{2k'} = \{3\}$. For both assignments of job 1, a schedule with a maximum completion time of $3p/2$ results. Thus, $C_{\max}^{\mathrm{LPE}'}/C_{\max}^{*} = 3/2$ as required. □

Further attempts to improve the heuristic are also possible. For example, a third schedule could be generated by setting p_{1k} to be large and resetting p_{2k} to its original value and reapplying heuristic LPE, thereafter selecting the best of the three schedules generated. Further schedules could also be generated by adjusting the processing times of both job k and job k' before reapplying heuristic LPE. Many variations on this theme are possible. Unfortunately, the worst-case analysis of these further modifications does not appear so straightforward.

4. Concluding remarks

The linear programming heuristic has a computational requirement that is polynomial in n although it is exponential in m. When $m \geq 3$, it has a worst-case performance ratio of 2 which represents a substantial improvement on the heuristic methods of Ibarra and Kim [8] and Davis and Jaffe [2] which both have worst-case

performance ratios that are functions of m. Unfortunately, the method does not appear to generalize satisfactorily to problems in which jobs have release dates or deadlines, or to problems in which there are precedence constraints on the jobs. For these more general problems, there are too many constraints and consequently too many basic variables in the linear programming relaxation to guarantee that a sufficiently large number of jobs can be assigned to machines in the partial schedule.

When applied to two machines, the heuristic can be modified to give a worst-case performance ratio of 3/2 while requiring linear time. This represents an improvement on the previous most effective heuristic of Ibarra and Kim [8] which requires $O(n \log n)$ time and has a worst-case performance ratio of $(1+\sqrt{5})/2$. The idea that is used in Section 3 of applying a heuristic to a problem after which the problem is modified and the heuristic reapplied is a useful technique for obtaining a modified heuristic with a superior worst-case performance. The technique has been successfully applied to problems in which jobs have release dates and are either sequenced on a single machine [11] or have to pass through a two-machine flow-shop [12].

Acknowledgement

The author is grateful to B.J. Lageweg and J.K. Lenstra for useful discussions about the implementation of the heuristic for the case $m=2$.

References

[1] M. Blum, R.W. Floyd, V. Pratt, R.L. Rivest and R.E. Tarjan, Time bounds for selection, J. Comput. System Sci. 7 (1973) 448-461.

[2] E. Davis and J.M. Jaffe, Algorithms for scheduling tasks on unrelated processors, J. Assoc. Comput. Mach. 28 (1981) 721-736.

[3] M.L. Fisher, Worst-case analysis of heuristic algorithms, Management Sci. 26 (1980) 1-17.

[4] M.R. Garey, R.L. Graham and D.S. Johnson, Performance guarantees for scheduling algorithms, Oper. Res. 26 (1978) 3-21.

[5] T. Gonzalez, E.L. Lawler and S. Sahni, Optimal preemptive scheduling of two unrelated parallel processors in linear time, J. Assoc. Comput. Mach., to appear.

[6] G. Hadley, Linear Programming (Addison-Wesley, Reading, MA, 1962).

[7] E. Horowitz and S. Sahni, Exact and approximate algorithms for scheduling nonidentical processors, J. Assoc. Comput. Mach. 23 (1976) 317-327.

[8] O.H. Ibarra and C.E. Kim, Heuristic algorithms for scheduling independent tasks on nonidentical processors, J. Assoc. Comput. Mach. 24 (1977) 280-289.

[9] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller and J.W. Thatcher, eds., Complexity of Computer Computations (Plenum Press, New York, 1972) 85-103.

[10] L.G. Khachian, A polynomial time algorithm in linear programming, Soviet Math. Dokl. 20 (1979) 191-194.

[11] C.N. Potts, Analysis of a heuristic for one machine sequencing with release dates and delivery times, Oper. Res. 28 (1980) 1436-1441.

[12] C.N. Potts, Analysis of heuristics for two-machine flow-shop sequencing subject to release dates, Report BW 150, Mathematisch Centrum, Amsterdam, 1981.

[13] A. Schonhage, M. Paterson and M. Pippenger, Finding the median, J. Comput. Systems Sci. 13 (1976) 189-199.