ALGORITHMS FOR SCHEDULING INDEPENDENT JOBS WITH RESTRICTED PROCESSING TIMES

Jayme Luiz Szwarcfiter

NCE 0386

April, 1986

Universidade Federal de Rio de Janeiro
Núcleo de Computação Eletrônica
Caixa Postal 2324
20001 - Rio de Janeiro - RJ

ALGORITHMS FOR SCHEDULING INDEPENDENT FOR WITH RESTRICTED PROCESSING TIMES

Jayme Luiz Szwarcfiter

ABSTRACT

We describe exact algorithms for $R//C_{max}$, $P//C_{max}$ and $RM//C_{max}$. Let k be the maximum cardinatily of a subset of jobs, any two of them with different processing times in some machine If k is fixed the algorithms terminate within polynomial time. In this case, if additionally the maximum processing time can be expressed as a polynomial in the number n of jobs then $Pm//C_{max}$ can be solved in 0 (n) time. The proposed algorithms allow the processing times to be real numbers, except that for $Pm//C_{max}$ which restricts them to integers.

RESUMO

Descrevemos algoritmos para $R//C_{max}$, $P//C_{max}$ e $Rm//C_{max}$. Seja n a menor cardinalidade de um subconjunto de tarefas, onde duas quaisquer dessas possuem tempos de processamento diferentes em alguma máquina. Se k é fixo os algoritmos terminam em tempo polinomial. Nesse caso, se adicionalmente o tempo de processamento máximo for expresso como um polinômio no número n de tarefas então $Rm//C_{max}$ pode ser resolvido em tempo 0(n). Os algoritmos propostos permitem que os tempos de processamento sejam números reais, exceto no problema $Rm//C_{max}$, que os restringe a inteiros.

1. INTRODUCTION

Throughout this paper $J=\{J_1,\ldots,J_n\}$ denotes a set of independent jobs and $M=\{M_1,\ldots,M_m\}$ unrelated machines, $n,m > 0$. J_i requires an arbitrary real (unless otherwise stated) processing time $p_{ij} \geq 0$ in M_j, $1\leq i\leq n$ and $1\leq j\leq m$. Each job is to be assigned non preemptively to any one of the machines.

We consider the problems of finding a minimum length schedule for the jobs of J in the following three cases: (i) there is an arbitrary number of unrelated machines, (ii) arbitrary number of identical (parallel) machines and (iii) fixed number of unrelated ones. These problems are all NP-hard [1] and in terms of the notation [2] correspond to $R//C_{max}$, $P//C_{max}$ and $Rm//C_{max}$, respectively. We describe exact algorithms for each of the cases. Problems (i) and (ii) are solved by dynamic programming and the corresponding algorithms allow the processing times to be real numbers. The solution for $Rm//C_{max}$ instead employs integer linear programming (ILP) and restricts the p_{ij}'s to be integers.

Let k be the maximum cardinality of a subset of jobs, any two of them have different processing times in some machine. The complexities of the proposed algorithms are exponentials in k, but not necessarily in n. If k is fixed all terminate within polynomial time. The algorithm of Leung [4] for $P//C_{max}$ has such a similar property.

In Section 2 we describe the algorithm for $R//C_{max}$, which has complexity of time $O(n^{2k}\ m)$ and space $O(n^k\ m)$. Section 3 formulates the algorithm for $P//C_{max}$ which requires $O(n^{2k} \log m)$ time and $O(n^k \log m)$ space. Finally the last section presents a method for solving $Rm//C_{max}$ requiring $O(n + 2^{mk}(k^2\log(n + p_{max}))c^{mk})$

time, $p_{max} = \max\ \{p_{ij}\}$. Clearly, the latter reduces to $0(n)$ for fixed k and $p_{max} = 0(n^b)$, with b,c constants.

The algorithm of Leung also employs dynamic programming for solving $P//C_{max}$, however restricting the processing times to integers. It has time complexity $0(n^{2(k-1)} \log m \log p_{max})$ and space $0(n^{k-1} \log m)$. Therefore a comparison of its time bound with that of the proposed method would depend on the relative values of n and p_{max}. It should be noted that the algorithm [4] can be extended to solve $R//C_{max}$, if we maintain the processing times as integers. In this case, its time and space complexities become $0(n^{2(k-1)}\ m \log p_{max})$ and $0(n^{k-1}\ m)$, respectively.

Finally, if j is a non negative integer let $Z_j=\{0,\ldots,j\}$ and $Z_j^+=\{1,\ldots,j\}$.

2. ARBITRARY NUMBER OF UNRELATED MACHINES

Let π_0 be a scheduling problem consisting of the jobs of J and unrelated machines M. In this section is described an algorithm for finding a schedule for π_0 having minimum length C_{max}.

We employ dynamic programming. If m = 1 the solution is simple. Otherwise π_0 is decomposed into the subproblems π_1 and π_2. The first has jobs $J(\pi_1) \subset J$ and machines $\{M_1,\ldots,M_{m-1}\}$, while π_2 consists of $J-J(\pi_1)$ and $\{M_m\}$, respectively. Let λ denote the minimum value of C_{max} for a subproblem. Clearly,

$$\lambda(\pi_0) = \min_{\pi_1,\pi_2} \{\max\ \{\lambda(\pi_1),\ \lambda(\pi_2)\}\}$$

Let $L_1 \cup \ldots \cup L_k = J$ be a partition of J into non empty disjoint subsets L_i, called <u>classes</u> of π_0, such that two jobs $J_a, J_b \in J$ belong to the same class iff

$$p_{aj} = p_{bj}, \quad 1 \le j \le m \tag{1}$$

Define $p^*_{ij} := p_{aj}$, $1 \le i \le k$ and $1 \le j \le m$, where J_a is any job belonging to L_i. The values p^*_{ij}, $1 \le j \le m$, are the <u>class processing times</u> of L_i.

Let π be a subproblem with jobs $J(\pi) \subset J$. The <u>profile</u> of π is the k-sequence $F = \langle f_1, \ldots, f_k \rangle$ such that $f_i = |L_i \cap J(\pi)|$, $1 \le i \le k$. $S = \langle s_1, \ldots, s_k \rangle$ is a <u>subprofile</u> of π when each $s_i \in Z_{f_i}$. Let $S' = \langle s_i' \rangle$ and $S'' = \langle s_i'' \rangle$ be subprofiles satisfying $s_i' \ge s_i''$, $1 \le i \le k$. Then $S'-S''$ denotes the subprofile $\langle s_1'-s_1'', \ldots, s_k'-s_k'' \rangle$.

Denote by $\lambda(F,j)$ the minimum value of C_{max} for a subproblem π having profile F and machines $\{M_1, \ldots, M_j\}$, $j \in Z^+_m$. By $\lambda^+(F,j)$ represent the corresponding minimum value as above, except that there is a single machine $\{M_j\}$, instead of $\{M_1, \ldots, M_j\}$. Finally, denote by $S^*(\pi)$ the set of all subprofiles of π.

The single machine problems can be solved directly:

$$\lambda^+(F,j) := \sum_{1 \le i \le k} f_i p^*_{ij} \tag{2}$$

The following recurrence relates the variables λ among subproblems.

$$\lambda(F,j) := \min_{S \in S^*(\pi)} \{\max\{\lambda(F-S,j-1), \lambda^+(S,j)\}\},$$

$$1 \le j \le m \text{ and } F \in S^*(\pi_0) \tag{3}$$

boundary condition

$$\lambda(F,0) := 0, \text{ for any } F. \tag{4}$$

The algorithm can now be described. The input to π_0 is the n x m matrix of processing times p_{ij}. Start by finding the classes and profile $F(\pi_0)$. Then for each subprofile of π_0 compute (2) and afterwards (3)-(4). The process terminates when $\lambda(F(\pi_0),m)$ is evaluated.

The construction of the actual minimum schedule can be done by tracing back the minimizing values of $S \in S^*(\pi_0)$.

The classes can be determined in $0(n^2m)$ time by computing (1) for each pair of jobs. This complexity can be improved to $0(nkm)$ by applying bucket sort techniques.

Generating the set of subprofiles of a subproblem π is equivalent to finding all distributions of at most $|J(\pi)|$ identical objects into k distinct cells, such that there are no more than f_i objects in the i-th cell, $1 \le i \le k$, where $<f_1,...,f_k>$ is the profile of π. These distributions can be easily obtained in lexicographical order, for instance.

There are $0(n^k)$ subprofiles of π_0 and each requires the computation of all subprofiles of its corresponding subproblems. The complexities of finding the classes and computing (2) are dominated by that of (3)-(4). Therefore the algorithm for $R//C_{max}$ requires $0(mn^{2k})$ time and $0(n^km)$ space.

The correctness of (2) follows from a trivial counting, while that of (3)-(4) is based on the decomposition described.

3. ARBITRARY NUMBER OF IDENTICAL MACHINES

In this section we consider the scheduling problem π_0

with jobs J and m parallel identical machines, i.e. $p_{ij} = p_{i\ell}$, for any j, $\ell \in Z_m^+$. Define $p_i := p_{ij}$, $1 \le i \le n$. We describe a variation of the algorithm of the previous section for finding a minimum length schedule for π_0.

As before, if m > 1 then π_0 is decomposed into two subproblems π_1 and π_2, having jobs $J(\pi_1) \subset J$ and $J-J(\pi_1)$, respectively. However, the number of machines of these subproblems is now made as equal as possible, i.e. $\lceil m/2 \rceil$ and $\lfloor m/2 \rfloor$, respectively. This will decrease the number of iterations needed to compute the recurrences.

Let $\alpha(m) \subset Z_m^+$ be the subset of integers constructed as follows.

$$\alpha(1) := \phi \tag{5}$$

$$\alpha(m) := \{m\} \cup \alpha(\lceil m/2 \rceil) \cup \alpha(\lfloor m/2 \rfloor), \text{ if } m > 1 \tag{6}$$

Clearly, $\alpha(m)$ is precisely the set of all possible number of machines of the subproblems generated by succesively applying the above decomposition. Also, $|\alpha(m)| = O(\log m)$.

Next, partition J into classes $L_1, \ldots, L_k$, as before in (1). Clearly, two jobs belong to the same class iff they have identical processing times. Define $p_i^* := p_a$, $1 \le i \le k$, $J_a \in L_i$.

Denote by $\lambda(F,j)$ the minimum value of C_{max} for a subproblem π having profile $F = \langle f_1, \ldots, f_k \rangle$ and j parallel identical machines.

The recurrences now become.

$$\lambda(F,1) := \sum_{1\le i\le k} f_i p_i^* \quad (7)$$

$$\lambda(F,j) := \min_{S\in S^*(\pi)} \{\max\{\lambda((F-S), \lceil j/2\rceil), \lambda(S,\lfloor j/2\rfloor)\}\},$$
$$j \in \alpha(m)-\{1\} \text{ and } F \in S^*(\pi_0) \quad (8)$$

The algorithm is now clear. Given π_0, find its classes, construct the profile $F(\pi_0)$ and using (5)-(6) obtain subset $\alpha(m)$. Then for each $j \in \alpha(m)$ in increasing order and each subprofile of π_0, compute (7)-(8). The computation stops when $\lambda(F(\pi_0),m)$ is calculated.

The set $\alpha(m)$ can be computed in $O(\log m)$ time. Using similar arguments as in Section 1 we conclude that the described algorithm for $P//C_{max}$ requires $O(n^{2k} \log m)$ time and $O(n^k m)$ space.

4. FIXED NUMBER OF UNRELATED MACHINES

Let π_0 be a scheduling problem with jobs J and unrelated machines M. We now describe an algorithm for minimizing the length of a schedule for π_0, using ILP. In this section, the processing times are restricted to integers.

Let $L_1,\ldots,L_k$ be the classes of π_0 obtained as in (1) and p_{ij}^*, $1\le j\le m$, the class processing times of L_i, $1\le i\le k$.

In any schedule of π_0, let x_{ij} be the number of jobs of class L_i assigned to machine M_j, $1\le i\le k$ and $1\le j\le m$. Let M_h be a machine in which its last job is completed at time C_{max}. The following is an ILP formulation for solving $Rm//C_{max}$.

$$\text{minimize} \sum_{1\le i\le k} x_{ih}\, p^*_{ih} \tag{9}$$

$$\text{s.t.} \quad \sum_{1\le i\le k} x_{ij}\, p^*_{ij} \le \sum_{1\le i\le k} x_{ih}\, p^*_{ih}, \quad \text{for } 1\le j\le m \tag{10}$$

$$\sum_{1\le j\le m} x_{ij} = |L_i|, \quad \text{for } 1\le i\le k \tag{11}$$

$$x_{ij} \text{ is an integer} \ge 0, \quad \text{for } 1\le i\le k,\ 1\le j\le m \tag{12}$$

Basically, (10) assures that C_{max} occurs in M_h, (11) that each job has been scheduled exactly once, while (9) is the minimization of the length of the schedule.

The algorithm is clear. For each h = 1,...,m solve the ILP problem (9)-(12). Then find the minimum among the minimizations (9).

The algorithm of Lenstra [3] solves an ILP problem with v variables and r constraints in $0(2^v (vr \log z)^c)$ time, where z is the largest coefficient in the problem. Therefore, the minimum length schedule for π_0 can be obtained in $0(nm + m2^{mk} (m^2k^2\log(n + p_{max}))^{c^{mk}})$ time, that is $0(n + (\log(n + p_{max}))^{c_1})$, for fixed m,k, where c, c_1 are constants.

REFERENCES

[1] M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-completeness (Freeman, San Francisco, 1979);

[2] R.L. Graham, E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan, Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey, Ann. Discrete Math. 5 (1979) 287-326;

[3] H.W. Lenstra Jr., Integer Programming with a Fixed Number of Variables, Mathematics of Oper. Research 8 (1983) 538-548;

[4] J.Leung, On Scheduling Independent Tasks with Restricted Execution Times, Oper. Research 30 (1982) 163-171.