Discrete Applied Mathematics 5 (1983) 11–24
North-Holland Publishing Company

SCHEDULING SUBJECT TO RESOURCE CONSTRAINTS: CLASSIFICATION AND COMPLEXITY

J. BLAZEWICZ

Politechnika Poznańska, Poznań, Poland

J.K. LENSTRA

Mathematisch Centrum, Amsterdam, The Netherlands

A.H.G. RINNOOY KAN

Erasmus University, Rotterdam, The Netherlands

Received 23 December 1981
Revised 6 April 1982

In deterministic sequencing and scheduling problems, jobs are to be processed on machines of limited capacity. We consider an extension of this class of problems, in which the jobs require the use of additional scarce resources during their execution. A classification scheme for resource constraints is proposed and the computational complexity of the extended problem class is investigated in terms of this classification. Models involving parallel machines, unit-time jobs and the maximum completion time criterion are studied in detail; other models are briefly discussed.

1. Introduction

In the traditional class of deterministic sequencing and scheduling problems [2, 7], *jobs* $J_1, \ldots, J_n$ consisting of one or more *operations* are to be processed on *machines* $M_1, \ldots, M_m$. Each machine can handle at most one job at a time and each job can be executed by at most one machine at a time. Thus, at any time, the execution of a job is restricted by the presence of a single scarce resource. We shall consider an extension of this class by allowing for the presence of more than one scarce resource. Each operation of a job requires the use of a given fraction of each of the resources, and the problem is to find an optimal schedule subject to these additional *resource constraints*. Such models occur for example in the context of computer operating systems and project scheduling.

Various assumptions can be made about the number of resources, about the amounts in which they are available, and about the amounts which are required by the operations. Section 2 introduces a simple *classification scheme* for resource constraints that captures many variations of the model. It expands the classification scheme for scheduling problems given in [7], the relevant part of which is included as an Appendix.

In general, the addition of resource constraints to a scheduling problem may affect its *computational complexity*. In particular, certain well-solved problems, for which polynomial-time algorithms exist, may be transformed into NP-hard ones, for which the existence of such algorithms is very unlikely [8, 5]. The obvious research program would be to determine the borderline between easy and hard resource constrained scheduling problems, much in the same vein as has been done for the traditional class, and possibly through the use of an extension of the computer aided complexity classification developed for that purpose [10]. Rather than attempting such a complete and probably somewhat tedious analysis, we will concentrate on single operation models with unit processing times and the maximum completion time criterion. Section 3 presents our results for these models. Section 4 deals briefly with some other models, *viz.* extensions to other optimality criteria, preemptive scheduling, and multi-operation models. Section 5 contains some concluding remarks.

2. Classification of resource constraints

Thee classification scheme for resource constrained scheduling problems introduced in [7] will be used in this paper as well. Briefly, a problem type corresponds to a three-field notation $\alpha|\beta|\gamma$, where α specifies the *machine environment*, β indicates certain *job characteristics*, and γ denotes the *optimality criterion*. Readers not familiar with this notation are referred to the Appendix, where all the relevant definitions can be found.

We shall expand this classification scheme by allowing the jobs to require the use of additional scarce resources. Suppose that there are *l resources* $R_1, \ldots, R_l$. For each resource R_h, there is a positive integer *size* s_h which is the total amount of R_h available at any given time. In single-operation models, there is for each resource R_h and job J_j a nonnegative integer *requirement* r_{hj} which is the amount of R_h required by J_j at all times during its execution. A schedule is *feasible* with respect to the resource constraints if at any time t the index set S_t of jobs being executed at t satisfies $\sum_{j\in S_t} r_{hj} \leq s_h$ $(h=1,\ldots,l)$. In multi-operation models, there is for each resource R_h and operation O_{ij} a nonnegative integer requirement r_{hij}, with a similar condition for the feasibility of a schedule.

The presence of scarce resources will be indicated in the second field of our classification scheme by

$$res\lambda\sigma\varrho$$

where λ, σ and ϱ are characterized as follows.

– If λ is a positive integer, then the *number of resources* l is constant and equal to λ; if $\lambda = \cdot$, then l is part of the input.

– If σ is a positive integer, then all *resource sizes* s_h are constant and equal to σ; if $\sigma = \cdot$, then all s_h are part of the input.

– If ϱ is a positive integer, then all *resource requirements* r_{hj} (r_{hij}) have a constant upper bound equal to ϱ; if $\varrho = \cdot$, then no such bounds are specified.

Many types of resource constraints are not represented by this classification, but in a sense more than enough detail is included already. In fact, we shall assume that λ, σ and ϱ are either equal to 1 or to $\cdot$; this restriction still generates most of the relevant and previously studied problem types.

Remembering that $\sigma = 1$ excludes $\varrho = \cdot$, we obtain six types of resource constraints, some of which are obvious generalizations of others. Fig. 1 illustrates these six types and the simple transformations between them; an arc from type (a) to type (b) indicates that (a) is a special case of (b).

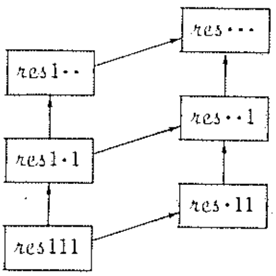


Fig. 1. Reductions between six types of resource constraints.

We can draw an additional arc from *res* ··· to *res*1·· under the restriction that the machines and resources are all *saturated* in each feasible schedule, i.e., $|S_t| = m$ and $\sum_{j \in S_t} r_{hj} = s_h$ $(h = 1, \ldots, l)$ at any time t until a given deadline. In this case the l requirements $r_{1j}, \ldots, r_{lj}$ can be encoded into a single mixed radix number r'_{1j} [4].

3. Single-operation models with unit processing times and the $C_{\max}$ criterion

We will now investigate the computational complexity of models involving parallel *identical* or *uniform* machines, unit-time jobs, (possibly empty) precedence constraints and the maximum completion time criterion. Theorems 1 to 7 determine the complexity of all such problems; the complete picture is given in Fig. 2.

Our starting point is the observation that a polynomial algorithm exists for the case of two identical machines, even under the most general type of resource constraints.

Theorem 1 (Garey & Johnson [4]). *$P2 \mid res\cdots, p_j = 1 \mid C_{\max}$ is solvable in* $O(ln^2 + n^{5/2})$ *time.*

Proof. Given any instance of $P2 \mid res\cdots, p_j = 1 \mid C_{\max}$, construct a graph G with vertices $1, \ldots, n$ and edges $\{j, k\}$ whenever $r_{hj} + r_{hk} \le s_h$ $(h = 1, \ldots, l)$. Thus, the

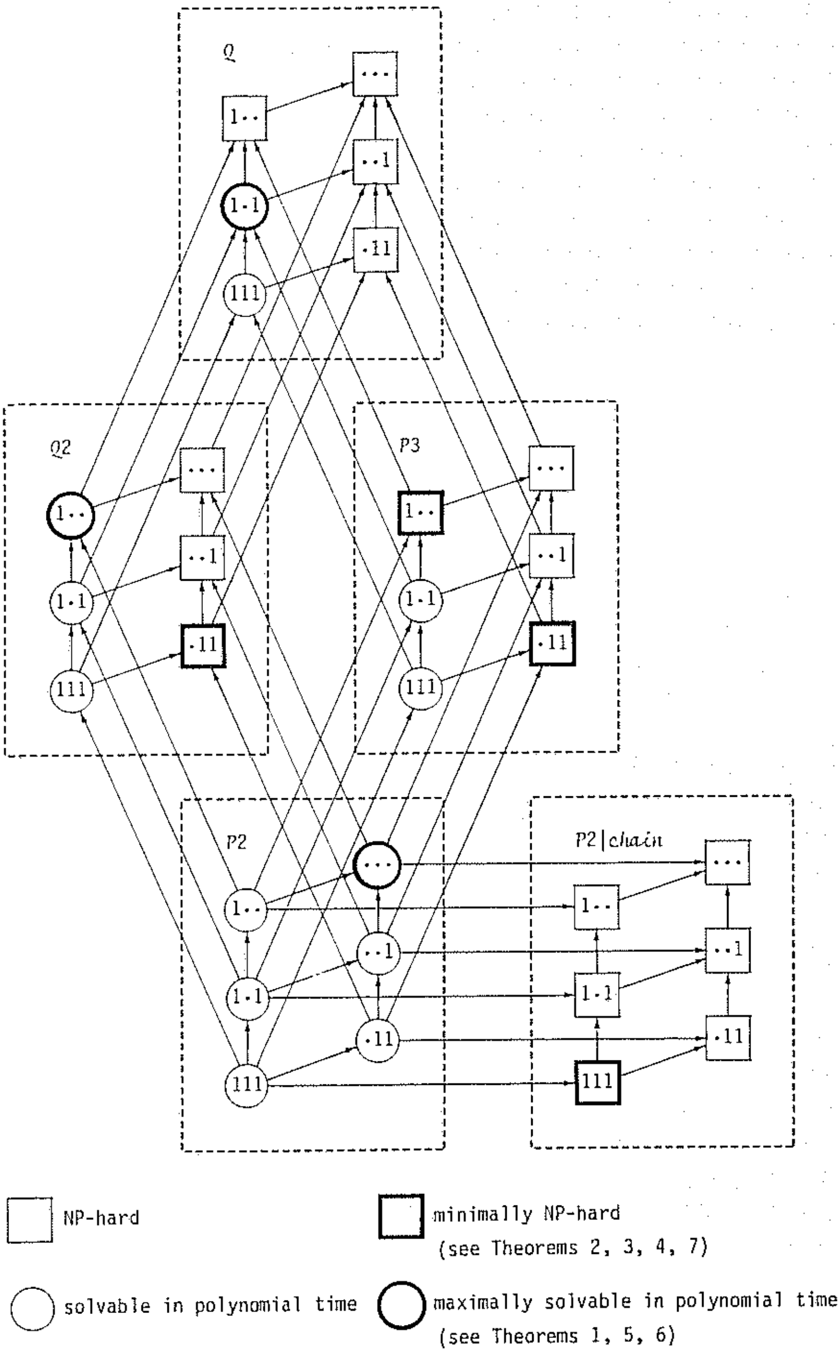


Fig. 2. Complexity of $\alpha|res\lambda\sigma\varrho, \beta_3, p_j=1|C_{max}$ problems.

vertices correspond to the jobs and the edges to pairs of jobs that can be executed simultaneously. Next, obtain a matching S (i.e., a set of vertex-disjoint edges) in G of maximum cardinality. Obviously, the minimum value of $C_{\max}$ is equal to $n-|S|$. Construction of G requires $O(ln^2)$ time, and the algorithm from [3] finds S in $O(n^{5/2})$ time. This proves the polynomial time bound. □

The correspondence between resource feasible sets of jobs and certain subsets of vertices in a graph can be turned around to obtain NP-hardness results for problems with three identical or two uniform machines. Given any graph G with vertex set V and edge set E, jobs and resource constraints of type $res\cdot 11$ can be defined in the following way:

– for each vertex $j\in V$, introduce a job J_j;

– for each vertex pair $\{j,k\}\notin E$, introduce a resource $R_{\{j,k\}}$ of size $s_{\{j,k\}}=1$ with requirements $r_{\{j,k\},j}=r_{\{j,k\},k}=1$, $r_{\{j,k\},i}=0$ otherwise.

Thus, two jobs can be executed simultaneously if and only if the corresponding vertices are adjacent.

Theorem 2. *$P3\,|\,res\cdot 11, p_j=1\,|\,C_{\max}$ is NP-hard in the strong sense.*

Proof. We present a straightforward transformation from the following NP-complete problem [5]:

PARTITION INTO TRIANGLES: Given a graph $G=(V,E)$ with $|V|=3t$, can V be partitioned into t disjoint subsets, each containing three pairwise adjacent vertices?

Given any instance of this problem, we construct an instance of $P3\,|\,res\cdot 11$, $p_j=1\,|\,C_{\max}$ in the way indicated above. Clearly, PARTITION INTO TRIANGLES has a solution if and only if there exists a feasible schedule with value $C_{\max}\le t$. □

Theorem 3. *$Q2\,|\,res\cdot 11, p_j=1\,|\,C_{\max}$ is NP-hard in the strong sense.*

Proof. In this case, we start from the following NP-complete problem [5]:

PARTITION INTO PATHS OF LENGTH 2: Given a graph $G=(V,E)$ with $|V|=3t$, can V be partitioned into t disjoint subsets, each containing three vertices, at most two of which are nonadjacent?

Given any instance of this problem, we construct an instance of $Q2\,|\,res\cdot 11$, $p_j=1\,|\,C_{\max}$ in the way indicated above, with machine speeds $q_1=2$, $q_2=1$. It is easily seen that PARTITION INTO PATHS OF LENGTH 2 has a solution if and only if there exists a feasible schedule with value $C_{\max}\le t$. □

Theorems 1, 2 and 3 indicate that, when there are no precedence constraints, we can restrict our attention to the case of a single resource. First, we recall a classical NP-hardness result.

Theorem 4 (Garey & Johnson [4]). *$P3 \mid res1 \cdot\cdot, p_j = 1 \mid C_{\max}$ is NP-hard in the strong sense.*

Proof. When the machines and resources are all saturated, $P3 \mid res1 \cdot\cdot, p_j = 1 \mid C_{\max}$ is equivalent to the following problem:

3-PARTITION: Given a set $S = \{1, \ldots, 3t\}$ and positive integers $a_1, \ldots, a_{3t}, b$ with $\sum_{j \in S} a_j = tb$, can S be partitioned into t disjoint 3-element subsets S_i such that $\sum_{j \in S_i} a_j = b$ $(i = 1, \ldots, t)$?

This celebrated problem was the first number problem proved to be NP-complete in the strong sense. □

It turns out that polynomial algorithms exist for all special cases of $Q \mid res \cdots, p_j = 1 \mid C_{\max}$ whose complexity status has not been settled so far. The solution methods are presented in Theorems 5 and 6.

Theorem 5. *$Q2 \mid res1 \cdot\cdot, p_j = 1 \mid C_{\max}$ is solvable in $O(n \log n)$ time.*

Proof. Given any instance of $Q2 \mid res1 \cdot\cdot, p_j = 1 \mid C_{\max}$, an optimal schedule can be obtained in the following way. Suppose that $q_1 \geq q_2$. First, schedule all jobs on M_1 in order of nonincreasing resource requirement. Next, successively remove the last job from M_1 and schedule it as early as possible on M_2, as long as this reduces the value of $C_{\max}$.

This $O(n \log n)$ algorithm clearly generates the best schedule among those satisfying the following properties:

(a) the jobs J_j on M_1 are executed in order of nonincreasing r_{1j} without machine idle time;

(b) the jobs J_k on M_2 are executed in order of nondecreasing r_{1k};

(c) $r_{1j} \geq r_{1k}$ for all J_j on M_1 and all J_k on M_2.

The correctness of the algorithm will now be proved by showing that any feasible schedule can be transformed into a schedule that is at least as good and satisfies properties (a), (b) and (c).

To avoid the introduction of some cumbersome notation, the transformation is presented in an informal way. Starting from a feasible initial schedule, one proceeds as follows (cf. Fig. 3).

Step 1. Move the jobs that are executed on M_2 while M_1 is idle to M_1. Interchange parts of the schedule simultaneously on both machines such that the jobs or fractions of jobs that are executed on M_1 while M_2 is idle are in the first positions on M_1.

Step 2. Interchange parts of the schedule simultaneously on both machines such that all jobs J_k on M_2 are in order of nondecreasing r_{1k}.

Step 3. Rearrange the (fractional) jobs J_j that are executed on M_1 while M_2 is busy in such a way that they are in order of nonincreasing r_{1j} and the preemptions created by Step 2 are eliminated. (This does not lead to resource infeasibility.)

Step 4. Insert the (fractional) jobs that are executed on M_1 while M_2 is idle in

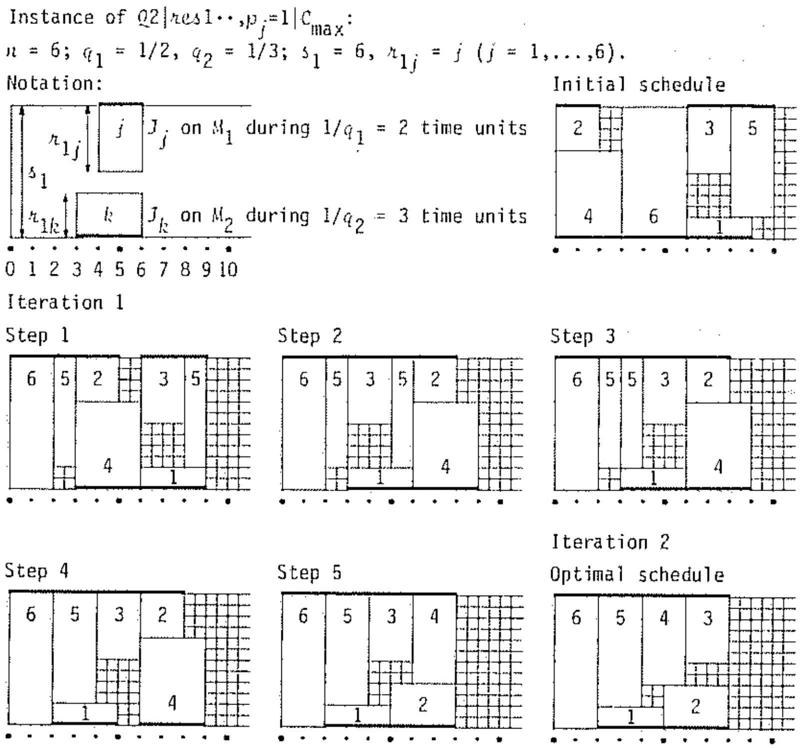


Fig. 3. Illustration of transformation of a $Q2\,|\,res1\cdot\cdot,\ p_j=1\,|\,C_{\max}$ schedule.

positions on M_1 chosen in such a way that all jobs J_j on M_1 are in order of nonincreasing r_{1j} and the preemptions created by Step 1 are eliminated; it may be necessary to introduce periods of idle time on M_2. Left-justify the resulting schedule.

Step 5. Let J_j be the last job on M_1 and J_k the last job on M_2. If $r_{1j} \geq r_{1k}$, the transformation terminates. Otherwise, schedule J_j in the position of J_k on M_2, schedule J_k as early as possible on M_1, left-justify the schedule, and return to Step 1.

None of these steps increases the value of $C_{\max}$. After each application of Steps 1 to 4, properties (a) and (b) are satisfied, and after a finite number of applications of Step 5, property (c) holds as well. This validates the algorithm given above. □

Theorem 6. *$Q\,|\,res1\cdot 1,\ p_j=1\,|\,C_{\max}$ is solvable in $O(n^3)$ time.*

Proof. Given any instance of $Q\,|\,res1\cdot 1,\ p_j=1\,|\,C_{\max}$, construct a transportation network with n sources j $(j=1,\ldots,n)$ and mn sinks (i,k) $(i=1,\ldots,m;\ k=1,\ldots,n)$. Each arc $(j,(i,k))$ has a cost c_{ijk}, to be defined below. The arc flow x_{ijk} is to have the following interpretation:

$$x_{ijk}=\begin{cases}1 & \text{if } J_j \text{ is executed on } M_i \text{ in the } k\text{th position,}\\ 0 & \text{otherwise.}\end{cases}$$

The number of resource requiring jobs executed simultaneously must never be allowed to exceed the resouce size. This can be effectuated by requiring that these jobs are assigned only to the fastest s_1 machines. Thus, assume that $q_h \geq q_i$ for all $h=1,\dots,s_1$ and all $i=s_1+1,\dots,m$, and define

$$c_{ijk}=\begin{cases}\infty & \text{if } i\geq s_1+1 \text{ and } r_{1j}=1,\\ k/q_i & \text{otherwise}\end{cases}$$

Then the problem is to minimize

$$\max_{i,j,k}\{c_{ijk}x_{ijk}\}$$

subject to

$$\sum_{i=1}^{m}\sum_{k=1}^{m} x_{ijk}=1 \quad (j=1,\dots,n),$$

$$\sum_{j=1}^{n} x_{ijk}\leq 1 \quad (i=1,\dots,m;\ k=1,\dots,n),$$

$$x_{ijk}\geq 0 \quad (i=1,\dots,m;\ j=1,\dots,n;\ k=1,\dots,n).$$

This bottleneck transportation problem can be formulated and solved in $O(n^3)$ time. □

Note. Similar transportation network models provide efficient solution methods for $Q\,|\,res1\cdot 1,\ p_i=1\,|\,\gamma$, where $\gamma\in\{\max_j\{f_j(C_j),\sum_j f_j(C_j)\}$ for arbitrary non-decreasing cost functions f_j $(j=1,\dots,n)$.

When the presence of precedence constraints between the jobs is allowed, NP-hardness in the strong sense has been established for $P2\,|\,res1\cdot\cdot,\ tree,\ p_j=1\,|\,C_{\max}$ [4] and $P2\,|\,res111,\ prec,\ p_j=1\,|\,C_{\max}$ [12]. These results are both dominated by Theorem 7.

Theorem 7. *$P2\,|\,res111,\ chain,\ p_j=1\,|\,C_{\max}$ is NP-hard in the strong sense.*

Proof. We prove this result by means of a transformation from 3-PARTITION (see Theorem 4), where we assume without loss of generality that $\frac{1}{4}b<a_j<\frac{1}{2}b$ for all $j\in S$. Given any instance of this problem, we construct an instance of $P2\,|\,res111$, *chain*, $p_j=1\,|\,C_{\max}$ in the following way:

– There is a single chain L of $2tb$ jobs:

$$\begin{aligned}L={}&J'_1\to J'_2\to\cdots\to J'_b\to J_1\to J_2\to\cdots\to J_b\to\\ &\to J'_{b+1}\to J'_{b+2}\to\cdots\to J'_{2b}\to J_{b+1}\to J_{b+2}\to\cdots\to J_{2b}\to\\ &\cdots\\ &\to J'_{(t-1)b+1}\to J'_{(t-1)b+2}\to\cdots\to J'_{tb}\to J_{(t-1)b+1}\to J_{(t-1)b+2}\to\cdots\to J_{tb}.\end{aligned}$$

– For each $j \in S$, there are two chains K_j and K_j', each of a_j jobs:

$$K_j = J_{j1} \to J_{j2} \to \cdots \to J_{ja_j},$$

$$K_j' = J_{j1}' \to J_{j2}' \to \cdots \to J_{ja_j}';$$

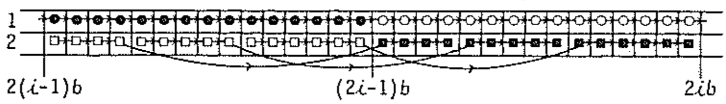
moreover, it is required that K_j precedes K_j', i.e., $J_{ja_j} \to J_{j1}'$.

– The primed jobs do require the resource, the unprimed jobs do not.

We claim that 3-PARTITION has a solution if and only if there exists a feasible schedule with value $C_{\max} \le 2tb$.

Suppose that 3-PARTITION has a solution $\{S_1, \dots, S_t\}$. A feasible schedule with value $C_{\max} = 2tb$ is then obtained as follows (cf. Fig. 4). First, the chain L is scheduled on machine M_1 in the interval $[0, 2tb]$; note that this leaves the resource available only in the intervals $[(2i-1)b, 2ib]$ $(i = 1, \dots, t)$. For each $i \in \{1, \dots, t\}$, it is now possible to schedule the three chains K_j $(j \in S_i)$ on machine M_2 in the interval $[2(i-1)b, (2i-1)b]$ and the chains K_j' $(j \in S_i)$ on M_2 in $[(2i-1)b, 2ib]$. The resulting schedule is feasible with respect to resource and precedence constraints and has total length $2tb$.

Part of feasible instance for 3-PARTITION:
$b = 15$, $a_1 = 4$, $a_2 = 5$, $a_3 = 6$; $S_i = \{1,2,3\}$.
Part of feasible schedule for $P2|res111, chain, p_j = 1|C_{\max}$:



●: J_k', ○: J_k $((i-1)b+1 \le k \le ib)$; □: J_{jk}, ■: J_{jk}' $(j \in S_i,\ 1 \le k \le a_j)$.

Fig. 4. Illustration of transformation from 3-PARTITION to $P2|res111, chain, p_j = 1|C_{\max}$.

Conversely, suppose that there exists a feasible schedule with value $C_{\max} \le 2tb$. It is clear that in this schedule both machines and the resource are saturated until time $2tb$. Moreover, the chains K_j $(j \in S)$ are executed in the intervals $[2(i-1)b, (2i-1)b]$ $(i = 1, \dots, t)$ and the chains K_j' $(j \in S)$ in the remaining intervals. Let S_i be the index set of chains K_j completed in the interval $[2(i-1)b, (2i-1)b]$, for $i = 1, \dots, t$. Consider the set S_1. It is impossible that $\sum_{j \in S_1} a_j > b$, due to the definition of S_1; the case $\sum_{j \in S_1} a_j < b$ cannot occur either, since this would lead to machine idle time in $[b, 2b]$. It follows that $\sum_{j \in S_1} a_j = b$, and our assumption about the size of a_j $(j \in S)$ implies that $|S_1| = 3$. This argument is easily extended to an inductive proof that $\{S_1, \dots, S_t\}$ constitutes a solution to 3-PARTITION. □

4. Other models

We will next comment on the computational complexity of three variations of the models considered in the previous section, viz.

(1) extensions to other optimality criteria,

(2) preemptive scheduling, and

(3) multi-operation models.

4.1. *Other optimality criteria*

If the $C_{\max}$ criterion is replaced by other optimality criteria such as the *total completion time* $\sum C_j$ or the *maximum lateness* $L_{\max}$, most results can be extended in a straightforward way.

In fact, all the NP-hardness results of Theorems 2, 3, 4 and 7 carry over immediately to both $\sum C_j$ and $L_{\max}$. For $\sum C_j$, we use the fact that the machines are saturated in each of the transformations; e.g., in Theorems 2 and 4 we have $C_{\max} \le t$ if and only if $\sum C_j \le \frac{3}{2}t(t+1)$. For $L_{\max}$, we define due dates $d_j = 0$ for all jobs, so that $L_{\max} = C_{\max}$.

It has been noted already that the transportation network model of Theorem 6 provides polynomial algorithms for $Q \mid res1 \cdot 1,\ p_j = 1 \mid \gamma$, where $\gamma = \sum C_j$ or $\gamma = L_{\max}$. The matching approach of Theorem 1 is easily adapted to solve $P2 \mid res \cdots,$ $p_j = 1 \mid \sum C_j$ as well: simply schedule the paired jobs before the remaining ones. It seems a safe conjecture that the algorithm of Theorem 5 can be modified to solve $Q2 \mid res1 \cdot\cdot,\ p_j = 1 \mid \sum C_j$; we leave this as a challenge to the reader. However, $P2 \mid res \cdots,\ p_j = 1 \mid L_{\max}$ and $Q2 \mid res1 \cdot\cdot,\ p_j = 1 \mid L_{\max}$ remain open problems. We mention that $P \mid res1 \cdot 1,\ r_j,\ p_j = 1 \mid L_{\max}$, where the r_j denote integer *release dates* at which the jobs become available, is solvable in polynomial time [1].

4.2. *Preemptive scheduling*

If the processing times are arbitrary and preemption is allowed, the nature of the models changes considerably. It now becomes of interest to consider the general case of parallel *unrelated* machines.

The problem $R \mid pmtn,\ res \cdots \mid C_{\max}$ can be formulated as a linear program in the following way (cf. [13, 11]). First, introduce a dummy job J_0 with $r_{h0} = 0$ for $h = 1, \ldots, l$, representing machine idle time. Define S as the set of all resource feasible m-tuples $k = (k_1, \ldots, k_m)$ of job indices; each k is characterized by:

– $k_i \in \{0, 1, \ldots, n\}$ for $i = 1, \ldots, m$;

– each $j \in \{1, \ldots, n\}$ occurs at most once;

– $\sum_{i=0}^{m} r_{hk_i} \le s_h$ for $h = 1, \ldots, l$.

To each $k \in S$, associate a variable x_k, representing the time during which $J_{k_1}, \ldots, J_{k_m}$ are simultaneously executed on $M_1, \ldots, M_m$ respectively. Then the problem is to minimize

$$\sum_{k\in S} x_k$$

subject to

$$\sum_{i=1}^{m}\left(\sum_{k\in S,k_i=j} x_k\right)/p_{ij}=1 \quad (j=1,\ldots,n),$$

$$x_k\geq 0 \qquad (k\in S).$$

This linear programming problem has $O(n^m)$ variables. For a fixed number of machines, its size is bounded by a polynomial in the size of the scheduling problem. The existence of a polynomial algorithm for linear programming [9] therefore implies that $Rm \mid pmtn,\ res\cdots \mid C_{\max}$ is solvable in polynomial time.

For a variable number of machines, $Q \mid pmtn,\ res111 \mid C_{\max}$ can be solved as follows. Replace the resource requiring jobs by a single job with execution requirement $\sum_{r_{1j}=1} p_j$; this eliminates the resource constraints. Next, apply the $O(m \log m+n)$ algorithm for $Q \mid pmtn \mid C_{\max}$ from [6] to solve the resulting problem.

4.3. Multi-operation models

Multi-operation models, in which each operation has its own specific resource requirements, give rise to various interesting results and to many open problems. By way of example, we consider open shops, flow shops and job shops with two machines, nonpreemptable operations and the $C_{\max}$ criterion.

In the case of an *open shop*, $O2 \mid res\cdots,\ p_{ij}=1 \mid C_{\max}$ is solvable by a matching approach similar to the one used in Theorem 1. If the processing times are arbitrary, even $O2 \mid res111 \mid C_{\max}$ remains unresolved.

Flow shop problems seem to be more difficult. $F2 \mid res111,\ p_{ij}=1 \mid C_{\max}$ is solvable in linear time by appropriately grouping jobs together according to their overall resource requirements. Little can be said about the immediate extensions of this model with unit processing times, but $F2 \mid res111 \mid C_{\max}$ is NP-hard in the strong sense by virtue of a simple transformation from 3-PARTITION.

The simplest *job shop* model in this context, $J2 \mid res111,\ p_{ij}=1 \mid C_{\max}$, is already NP-hard in the strong sense; the transformation from 3-PARTITION is nontrivial.

5. Concluding remarks

We have proposed a classification scheme for resource constrained scheduling problems and outlined a range of initial results on their computational complexity. Presumably, many of the remaining open problems can be resolved along similar lines. We hope to have stimulated others to continue the investigation of this interesting research area.

Appendix: Classification of scheduling problems

Suppose that n *jobs* $J_1, \ldots, J_n$ have to be processed on m *machines* $M_1, \ldots, M_m$. Each machine can handle at most one job at a time and each job can be executed by at most one machine at a time. Various job, machine and scheduling characteristics are reflected by a three-field problem classification $\alpha \mid \beta \mid \gamma$ [7]. Let $\circ$ denote the empty symbol.

Machine environment

The first field $\alpha = \alpha_1\alpha_2$ specifies the machine environment.

If $\alpha_1 \in \{P, Q, R\}$, each J_j consists of a single operation that can be procesed on any M_i; the processing time of J_j on M_i is p_{ij} $(i=1, \ldots, m;\ j=1, \ldots, n)$. The three values are characterized as follows.

– $\alpha_1 = P$ (*parallel identical machines*): $p_{ij} = p_j$ for a given execution requirement p_j of J_j.

– $\alpha_1 = Q$ (*parallel uniform machines*): $p_{ij} = p_j/q_i$ for a given execution requirement p_j of J_j and a given speed q_i of M_i.

– $\alpha_1 = R$ (*parallel unrelated machines*): p_{ij} is arbitrary.

If $\alpha_1 \in \{O, F, J\}$, each J_j consists of a set of m_j operations O_{ij}; O_{ij} has to be processed on a given machine μ_{ij} during p_{ij} time units $(i=1, \ldots, m_j;\ j=1, \ldots, n)$. The three values are characterized as follows.

– $\alpha_1 = O$ (*open shop*): $m_j = m$, $\mu_{ij} = M_i$.

– $\alpha_1 = F$ (*flow shop*): $m_j = m$, $\mu_{ij} = M_i$; $O_{i-1,j}$ has to be completed before O_{ij} can start $(i=2, \ldots, m)$.

– $\alpha_1 = J$ (*job shop*): m_j and μ_{ij} are arbitrary; $\mu_{i-1,j} \neq \mu_{ij}$ and $O_{i-1,j}$ has to be completed before O_{ij} can start $(i=2, \ldots, m_j)$.

If α_2 is a positive integer, then m is constant and equal to α_2; if α_2 is $\circ$, then m is part of the input.

Job characteristics

The second field $\beta \subset \{\beta_1, \beta_2, \beta_3, \beta_4\}$ indicates a number of job characteristics, which are defined as follows.

(1) $\beta_1 \in \{pmtn, \circ\}$.

– $\beta_1 = pmtn$: Preemption (job splitting) is allowed; the processing of any job may arbitrarily often be interrupted and resumed at the same time on a different machine or at a later time on any machine.

– $\beta_1 = \circ$: No preemption is allowed.

(2) β_2 specifies the resouce constraints; see Section 2.

(3) $\beta_3 \in \{prec, tree, chain, \circ\}$.

– $\beta_3 = prec$ (arbitrary precedence constraints): A directed acyclic graph H with vertices $1, \ldots, n$ is given; if H contains a directed path from j to k, we write $J_j \rightarrow J_k$

and require that J_j is completed before J_k can start.

– $\beta_3 = tree$ (tree-like precedence constraints): H has outdegree at most one for each vetex or indegree at most one for each vertex.

– $\beta_3 = chain$ (chain-like precedence constraints): H has both outdegree and indegree at most one for each vertex.

– $\beta_3 = \circ$ (no precedence constraints): H has no arcs.

(4) $\beta_4 \in \{p_{ij} = 1, \circ\}$.

– $\beta_4 = p_{ij} = 1$: Each operation has unit processing time.

– $\beta_4 = \circ$: The processing times are arbitrary nonnegative integers.

(If $\alpha_1 \in \{P, Q\}$, then p_{ij} is replaced by p_j; if $\alpha_1 = R$, then $\beta_4 = \circ$.)

Optimality criteria

The third field γ denotes the optimality criterion chosen. Any feasible schedule defines for each J_j a *completion time* C_j and, given an integer *due date* d_j, a *lateness* $L_j = C_j - d_j$ $(j = 1, \ldots, n)$. Some common optimality criteria involve the minimization of

– $C_{\max} = \max\{C_1, \ldots, C_n\}$ (*maximum completion time*);

– $\sum C_j = C_1 + \cdots + C_n$ (*total completion time*);

– $L_{\max} = \max\{L_1, \ldots, L_n\}$ (*maximum lateness*).

Acknowledgements

The authors gratefully acknowledge stimulating discussions with J. Weglarz. The research of the last two authors was partially supported by NATO Special Research Grant 9.2.02 (SRG.7) and by NSF Grant MCS78-20054.

References

[1] J. Blazewicz, Deadline scheduling of tasks with ready times and resource constraints, Information Processing Lett. 8 (1979) 60–63.

[2] R.W. Conway, W.L. Maxwell and L.W. Miller, Theory of Scheduling (Addison-Wesley, Reading, MA, 1967).

[3] S. Even and O. Kariv, An $O(n^{2.5})$ algorithm for maximum matching in general graphs, Proc. 16th Annual IEEE Symp. Foundations of Computer Science (1975) 100–112.

[4] M.R. Garey and D.S. Johnson, Complexity results for multiprocessor scheduling under resource constraints, SIAM J. Comput. 4 (1975) 397–411.

[5] M.R. Garey and D.S. Johnson, Computers and Intractability: a Guide to the Theory of NP-Completeness (Freeman, San Francisco, 1979).

[6] T. Gonzalez and S. Sahni, Preemptive scheduling of uniform processor systems, J. Assoc. Comput. Mach. 25 (1978) 92–101.

[7] 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.

[8] R.M. Karp, On the computational complexity of combinatorial problems, Networks 5 (1975) 45–68.

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

[10] B.J. Lageweg, E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan, Computer aided complexity classification of deterministic scheduling problems, Report BW 138, Mathematisch Centrum, Amsterdam (1981).

[11] R. Slowinski, Two approaches to problems of resource allocation among project activities – a comparative study, J. Operational Res. Soc. 31 (1980) 711–723.

[12] J.D. Ullman, Complexity of sequencing problems, in: E.G. Coffman, Jr., ed., Computer & Job/Shop Scheduling Theory (Wiley, New York, 1976) 139–164.

[13] J. Weglarz, J. Blazewicz, W. Cellary and R. Slowinski, Algorithm 520: an automatic revised simplex method for constrained resource network scheduling, ACM Trans. Math. Software 3 (1977) 295–300.