SIAM J. DISC. MATH.
Vol. 2, No. 4, pp. 550–581, November 1989

012

A MATHEMATICAL MODEL FOR PERIODIC SCHEDULING PROBLEMS*

PAOLO SERAFINI† AND WALTER UKOVICH‡

Abstract. A mathematical model is proposed for scheduling activities of periodic type. First a model is proposed for scheduling periodic events with particular time constraints. This problem, which could be considered the extension to periodic phenomena of ordinary scheduling with precedence constraints, is shown to be NP-complete. An algorithm for it of implicit enumeration type is designed based on network flow results, and its average complexity is discussed. Some extensions of the model are considered. The results of this first part serve as a basis in modelling periodic activities using resources. Several cases are considered. Finally some applications are presented for which the proposed model can be a useful tool.

Key words. scheduling, periodic scheduling, cyclic scheduling

AMS(MOS) subject classifications. 90B35, 90C35

1. Introduction.

1.1. Scope of the paper. This paper deals with the problem of scheduling periodic phenomena, that is, events and activities to be identically repeated at a constant rate. Periodic phenomena may arise either naturally, or as the consequence of imposed constraints for reasons of convenience or efficiency. In particular this may occur whenever a finite set of actions must be repeated with an infinite time horizon.

Examples of periodic phenomena are quite numerous and range over a wide spectrum of different applications involving problems of production, maintenance, transportation, vehicle scheduling, personnel scheduling, control, real-time processing, and so on.

The aim of this paper is to develop a consistent model into which some of the above problems might be conveniently approached both from the point of view of model clarity and from that of computational effort. We do not pretend to provide efficient and convenient methods to solve every kind of periodic scheduling problem, even if we think that the model proposed in this paper is sufficiently general to encompass several practical problems.

We realize that generality can be a drawback from the point of view of computational efficiency, which may be typically improved by avoiding generality and exploiting particular structures. Hence the computational approach we propose, while exhibiting a good average behaviour, should nevertheless be regarded as a departure point for the development of more sophisticated "ad hoc" exact and/or heuristic algorithms. In this sense, even if part of the paper is algorithmically oriented, its main concern is more about modelling issues.

A convenient way of defining and explaining the scope and the characteristics of our model consists in comparing it to classical (nonperiodic) "project scheduling" problems like the ones usually dealt with by CPM/PERT techniques. The ordinary ingredients of ordinary project scheduling are activities and their precedence constraints. The classical problem then consists in finding a feasible activity schedule minimizing the overall project completion time. This may be accomplished in polynomial time by dynamic programming algorithms working on an acyclic directed graph whose structure expresses the activities (nodes) and their precedence constraints (arcs).

* Received by the editors June 3, 1987; accepted for publication (in revised form) November 29, 1988.
† University of Udine, Dipartimento di Matematica e Informatica, Via Zanon 6, 33100 Udine, Italy.
‡ University of Trieste, Dipartimento di Elettrotecnica, Elettronica, e Informatica, Via Valerio 10, 34127 Trieste, Italy.

In the periodic version we are considering we deal with periodic activities with a given common period; the precedence constraints, which are meaningless in a periodic setting, are reformulated as "time window" constraints affecting the relative position of pairs of activities within the period.

Clearly there is no completion time to "minimize," and the period value, in the case of nonnatural periodicity, is a design parameter. We approach the problem of periodic scheduling by considering the period as a parameter given a priori. So we are primarily interested in finding feasible schedules with respect to the time window constraints and do not consider the problem of the best choice of the period value. If particular problems do exhibit natural objectives to minimize, then the model proposed here could be regarded as an initial formulation to be further refined.

A major difference between ordinary and periodic scheduling is given by their computational complexity. In fact, while ordinary project scheduling is polynomial, periodic scheduling is NP-complete, even if no resources are involved. On the other hand, the presence of resources makes ordinary project scheduling a qualitatively much harder problem, since it becomes NP-hard in most cases. Quite differently, the introduction of resources in periodic scheduling makes the problem only quantitatively more difficult, leaving it in the NP class. We remark that, in presence of resources, the same type of computational complexity appears both in ordinary and periodic scheduling. Therefore we think that our model could also benefit from the research carried out in ordinary scheduling in order to gain computational improvements.

1.2. A brief review of pertinent results. Despite its potential interest, papers on periodic scheduling constitute only a negligible fraction of the overall literature on scheduling problems (compare, for instance, [10], [17], [22]). Most of them are concerned with specific applications, such as cyclic staffing, production planning, etc. Both the models and the methods proposed for such cases are often strongly application-oriented, so they may bring little insight for a general approach.

A formulation of periodic scheduling problems with several similarities to our model is proposed by Dauscha, Modrow, and Neumann [8]. They deal with activities of fixed duration subject to constraints on the minimum time distance between their ordered activation times, thus giving rise to "cyclic precedence orders." As a mathematical tool they use the "cyclic sequence types" expressing the number of periods needed to carry out any given cyclic set of activities in a given order. Then they derive some interesting properties of the cyclic sequence types and a necessary and sufficient condition on them that guarantees the existence of a periodic schedule. A method is presented for constructing a periodic schedule, once a feasible cyclic sequence type is provided with time complexity bounded by the third power of the number of constraints; moreover, the overall problem of finding an admissible cyclic schedule is shown to be NP-complete and is formulated as a mixed-integer programming problem with Boolean variables. An application to a simple problem concerning the setting of a traffic light system operating periodically is thoroughly carried out.

Other aspects of periodic scheduling problems are considered by Kats [15]. He is concerned with finding periodic scheduling and routes for a system of interprocessor conveyors serving a production line. Processing and interprocessor transfer times are bounded to lie within prescribed intervals, and a minimum period schedule is sought. A result that allows the number of operators necessary along one route to be expressed as the number of periods necessary to travel through it is used to restrict the search for the optimal schedule to the solutions minimizing the number of operators in a given period. A branch-and-bound solution method is proposed, acting on the elements of the matrix

expressing the minimal number of periods required by a conveyor to perform two successive movements for a given schedule. It solves in polynomial time the case with fixed operation lengths, which corresponds to a resource minimization in our formulation.

A similar approach is considered by Orlin [18], who minimizes the number of vehicles to meet a fixed periodic schedule, with deadheading allowed. He formulates the problem by using a periodic version of the partially ordered sets and solves it in a polynomial time using a periodic version of the Dilworth's theorem, leading to results similar to the ones found in the next § 3.2, i.e., resource minimization in presence of a fixed activity schedule. The same problem may be reformulated as a coloring problem for a "periodic interval graph." A similar coloring problem, concerning "circular arc graphs" (see [28], [11]), corresponds again to the problem of assigning vehicles to a fixed periodic schedule, with the additional requirement that the same vehicle must always perform the same activity. As Orlin points out, although in general the latter problem is NP-hard, some simpler formulations may be solved in polynomial time as in Orlin, Bonuccelli, and Bovet [19].

Again similar problems are investigated by Gertsbakh and Gurevich [12], [13] in the framework of vehicle scheduling for a given time table by using the concept of "deficit function." Although their approach is entirely different from the ones previously outlined and from ours as well, the results they obtain are quite similar, as far as resource minimization is concerned. Along this line of research we may also quote a paper by Ceder and Stern [7].

An interesting class of periodic scheduling problems arising from production planning is the "economic lot scheduling problem." Its objective is finding an optimal schedule that allows cyclic production patterns for several products that are made on a single machine, in such a way that setup and inventory costs per unit time are minimized. A peculiar aspect of this problem is that different products may well have different frequencies of production, i.e., production periods of different lengths. Vemuganti [29] proposes a mixed-integer linear programming approach for this problem. Hsu [14] proves that this problem is NP-hard and presents an implicit enumeration scheme to test the feasibility of a given set of production periods. Problems with similar structure may also be approached by our model according to the results of §§ 2.5 and 3.4.

A similar class of problems, in which different periodic events may have different recurrence rates, is dealt with by Burkard [6]. He provides some interesting results for several cases, dealing with two or more events, with respect to different types of objective.

Another problem concerning periodic activities with different period lengths is considered by Park and Yun [21]. They seek to minimize the maximum amount of resources required by such activities. The problem is formulated as a 0-1 linear programming problem and the Chinese Remainder Theorem is used to decompose it.

A quite different kind of periodic scheduling problem concerns personnel scheduling. Here time is normally discretized in large intervals and the aim is usually to meet some given periodic workload pattern with weekly periodicity while trying to satisfy some additional requirements. Such problems originated a relatively large number of papers; just to quote a few of them, see the early paper by Baker [1] and the ones by Bartholdi, Orlin, and Ratliff [3], Bartholdi [2], Bechtold [5], Emmons [9]. Despite the interest of these types of problems and some apparent similarities with the previous problems, they are rather different and so we do not provide models for them, although it is perhaps possible to adapt our model to some of them.

An overall glance at the literature dealing with periodic scheduling reveals a wide spectrum of applications with a relatively more restricted range of approaches proposed to solve them: usually NP-completeness is shown for the general problem and an integer

(or mixed-integer) LP formulation is presented; then combinatorial heuristics for it or polynomial algorithms for simpler subproblems are proposed. Surprisingly enough, papers referring to different applications never consider the literature relative to other applications, even if they deal with abstract models that could be conveniently applied or extended to a larger class of problems. With this perspective, the present paper's objective is to provide a first unifying framework through abstract models apt to formalize basic features and phenomena that may be faced in several different applications.

1.3. Plan of the paper. The paper is structured in three main parts. In the first part (§ 2) the periodic event scheduling problem is defined with reference to time constraints only. Then in the second part (§ 3) resource constraints are taken into account, leading to more complex problems. In the third part (§ 4) some applications are described to which the previous models can be applied.

The detailed plan of the paper is as follows. The periodic event scheduling problem is first defined in terms of the abstract concept of a periodic event, its schedule, and a particular type of constraints for the schedule that affect the relative positions of pairs of periodic events. Then it is shown that the resulting problem is NP-complete.

The problem is reformulated as a network problem and an algorithm is derived from this formulation based on an implicit enumeration technique. The probabilistic performance of the algorithm is discussed and some computational evidence is provided.

Then the initial model is embedded with additional features, such as multiple periodicities and sequencing constraints, which turn out to be convenient when dealing with resources and therefore in several applications. At the conclusion of this part, a relationship with the triangular Traveling Salesman Problem is pointed out.

In the second part, activities and resources in a periodic setting are first defined and the peculiar structure of resource scheduling is investigated. As a consequence, the problem of scheduling periodic activities with resource constraints is reduced to the models developed in the previous part. In particular, the problem of minimizing the number of resource units is easily solved through a weighted assignment problem (obtaining a result similar to Bartlett [4], Orlin [18] and Gertsbakh and Gurevich [12], [13]), whereas different models of increasing complexity are presented for the problem of scheduling the activities with different types of resource constraints.

The last part is devoted to three applications. The first is a periodic version of the Job-Shop Problem. Similarities and dissimilarities between the periodic and the non-periodic version are investigated. A second application concerns the problem of vehicle scheduling according to a periodic time table. Finally, the problem of traffic light scheduling is outlined and a simplified version of it is modelled according to the previous results.

This paper is the refinement and extension of the preliminary results that already appeared in [24].

2. The periodic event scheduling problem.

2.1. Basic definitions. A *periodic event* ε is a countably infinite set of events $e(p)$, indexed by $p \in Z$, with occurrence times $t(e(p)) \in R$, such that for each p, $t(e(p)) - t(e(p-1)) = T$ (Z denotes the set of integers and R the set of reals). T is referred to as the *period* and $e(p)$ as the pth occurrence of the periodic event ε. We also define $t(\varepsilon) := t(e(p)) \bmod T$.

A periodic event ε is said to be *scheduled* if an element $\tau(\varepsilon) \in \Phi := R/T$ is associated to it, such that $\tau(\varepsilon) = \Pi \cdot t(e(p))$, for all p, where Π is the canonical projection from R to the abelian group Φ.

Let $D = [d^-, d^+]$ be any real closed interval. Then a *span* Δ of Φ is defined by $\Delta := \Pi \cdot D$. Note that Φ itself is a span by this definition; *proper spans* are defined by excluding the cases $\Delta = \Phi$ and $\Delta = \varnothing$.

Any proper span may be denoted by $\Delta = [\delta^-, \delta^+]$, with $\delta^- = \Pi \cdot d^-$ and $\delta^+ = \Pi \cdot d^+$, or, with abuse of notation, by $\Delta = [d^-, d^+]$, or even by $\Delta = [d^-, d^+]_T$ if it is necessary to specify the period. Note that $\Delta = [d^- + zT, d^+ + zT]_T$ for all $z \in Z$. The ambiguous case in which $\delta^- = \delta^+ = \delta$ will be proposed as $[\delta^-, \delta^+] = \{\delta\} = \Pi \cdot [d, d]$.

A *span constraint* a is a relationship involving an ordered pair of periodic events $(\varepsilon^-(a), \varepsilon^+(a))$ and a proper span $\Delta(a)$ and requiring that $\tau(\varepsilon^+(a)) - \tau(\varepsilon^-(a)) \in \Delta(a)$.

Loosely speaking, while nonperiodic events are arranged in a linear chronological order, periodic events have a cyclic chronological order and their occurrence times τ may be considered as points on a circumference which is scanned e.g., clockwise. A span constraint between two periodic events restricts the occurrence time of one event relative to the occurrence time of the other event to belonging to a certain time window on the circumference.

Throughout the paper we have tried as much as possible to employ a notation which should put in evidence the type of concepts with which we are dealing. So we use Greek letters (ε, τ, Δ, δ, etc.) to denote periodic or group quantities, and Latin letters (e, t, D, d, etc.) to denote either nonperiodic quantities or real representations of group quantities.

The Periodic Event Scheduling Problem (PESP) is the following one.

Given

— a finite set N of n periodic events with a common period T;

— a finite set A of span constraints;

find a *schedule* $\tau(\varepsilon) \in \Phi$, for all $\varepsilon \in N$, satisfying all span constraints.

For an example of a simple PESP instance refer to Fig. 1.1.

Multiple span constraints for the same pair of periodic events can also be imposed, thus allowing us to consider the intersection of the corresponding spans. However, a remarkable fact, due to the periodic structure, is that the intersection of two spans is not necessarily a span: in fact, it can be the union of two disjoint spans. As an example consider $[4, 5]_{10} \cup [7, 12]_{10} = [7, 15]_{10} \cap [4, 12]_{10}$ (see Fig. 1.2). Hence it is possible to model dichotomy constraints of the type $\tau(\varepsilon_2) - \tau(\varepsilon_1) \in \Delta_1 \cup \Delta_2$ by simply imposing

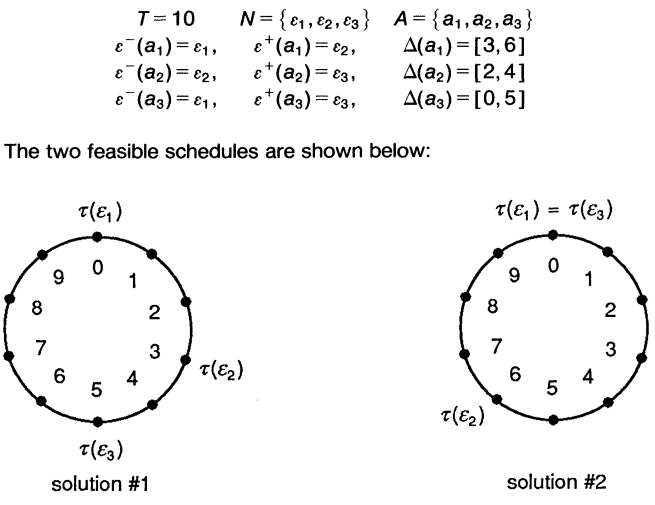

FIG. 1.1. *A* PESP *instance.*

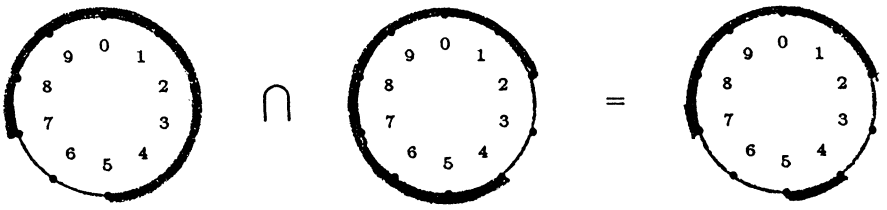

FIG. 1.2. *Intersection of spans.*

two appropriate span constraints for the same pair of periodic events. Of course, constraints involving the union of more than two spans can also be dealt with in a similar way.

2.2. Properties of PESP.

THEOREM 1. PESP *is* NP-*complete.*

Proof. It is easy to see that PESP is in NP. In order to prove NP-completeness, a polynomial transformation from the Hamiltonian Circuit Problem to PESP is produced. Let $G = (V, E)$ be a nondirected graph with n vertices for which a Hamiltonian circuit is sought. Let E' and F' be the arc sets of G and of the complete graph K_n, respectively, with arcs arbitrarily oriented. Then consider the following instance of PESP with $T = n$, N identified with V, A identified with F', $\Delta(a) = [1, n-1]$ if $a \in E'$ and $\Delta(a) = [2, n-2]$ otherwise.

Then PESP has a solution if and only if G has a Hamiltonian circuit. In fact, suppose a solution $\tau(\varepsilon)$ for PESP is produced: then $t(\varepsilon') - t(\varepsilon'') \bmod n \geqq 1$ for all $\varepsilon', \varepsilon'' \in N$, $\varepsilon' \neq \varepsilon''$, and there exists a cyclic permutation P on N such that $t(P(\varepsilon)) - t(\varepsilon) \bmod n = 1$ with ε and $P(\varepsilon)$, adjacent vertices in G.

Conversely, suppose that a cyclic permutation P is generated corresponding to a Hamiltonian circuit of G; it suffices to set $\tau(\varepsilon_1) = 0$ and $\tau(P(\varepsilon_i)) = \tau(\varepsilon_i) + 1$ in order to get a solution for PESP. □

It is immediate to check that PESP is also strongly NP-complete (see [20]).

It is useful to give a network representation of PESP. Given any instance of PESP, consider $G = (N, A)$ as a network: periodic events are considered as nodes and span constraints are considered as directed arcs. Furthermore, the following set of real spans is associated to each arc a

$$D(a, z) := [d^-(a) + zT,\ d^+(a) + zT] \quad \forall z \in Z$$

in such a way that $\Pi \cdot D(a, z) = \Delta(a)$.

For any network, a function $u : N \to R$ is called a *potential*, and the corresponding function $v : A \to R$ such that $v(a) = u(\varepsilon^+(a)) - u(\varepsilon^-(a))$ is called a *tension* (see [23]). In the context of the network representation of PESP, a span constraint is a constraint involving the tension of some arc a so that $v(a) \in D(a, z)$ for some integer z.

Then PESP is equivalent to the following network problem.

Find a potential u such that $v(a) \in D(a, z(a))$ for some $z(a) \in Z$, for all $a \in A$.

Figure 1.3 gives a network representation of the PESP instance of Fig. 1.1 and of its solution #1. Of course $\tau(\varepsilon) = \Pi \cdot u(\varepsilon)$.

If the resulting network turns out to be disconnected, then obviously the instance splits into a certain number of instances (one for each connected component) which can be solved independently. In view of this remark we shall always assume that the network is connected.

It is important to note that the above network problem reduces to the *Feasible Differential Problem* (FDP) if the values of the z's for all arcs are fixed a priori (see [23]). This fact will be exploited in designing an algorithm for PESP.

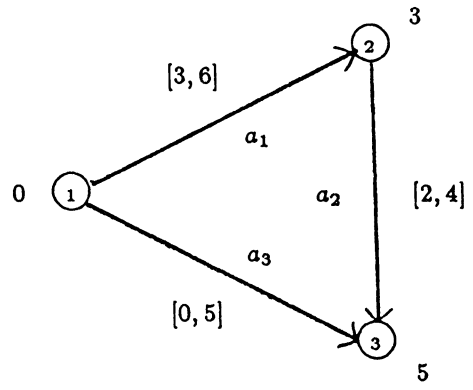

FIG. 1.3. *Network representation of* PESP.

The FDP is polynomial with complexity $O(n^3)$ and the difficulty for PESP comes in by allowing z to assume any integer value. It is interesting to point out some other differences between FDP and PESP.

For any circuit C, let C^+ and C^- be the set of arcs having the same and the opposite orientation, respectively, of the circuit C. Then define:

$$D(C) := \left[\sum_{a \in C^+} d^-(a) - \sum_{a \in C^-} d^+(a), \sum_{a \in C^+} d^+(a) - \sum_{a \in C^-} d^-(a) \right].$$

For instance, consider in Fig. 1.3 the unique circuit C oriented clockwise. Then $C^+ = \{a_1, a_2\}$, $C^- = \{a_3\}$ and $D(C) = [3 + 2 - 5, 6 + 4 - 0] = [0, 10]$.

An obvious necessary condition for PESP to have a solution is the following:

THEOREM 2. PESP *has a solution only if for any circuit* C *there exists* $z(C) \in Z : z(C)T \in D(C)$.

Proof. First note that there is no difference in reversing the orientation of some arc a and replacing its span $[d^-(a), d^+(a)]$ with $[-d^+(a), -d^-(a)]$. Now take any circuit C and reverse the orientation of the arcs in C^-. Given a feasible potential, the sum of the tensions along the circuit C, computed according to the new orientations, must be zero. Since $d^-(a) \leqq v(a) - z(a)T \leqq d^+(a)$ if $a \in C^+$ and $-d^+(a) \leqq v(a) + z(a)T \leqq -d^-(a)$ otherwise, the thesis follows by simply summing up the inequalities along the circuit. □

In the example of Fig. 1, this condition is satisfied with both $z = 0$ and $z = 1$. However, the condition is not sufficient. Consider the instance in Fig. 2 that satisfies the condition but is not feasible. This result of nonsufficiency was also expected by theoretical

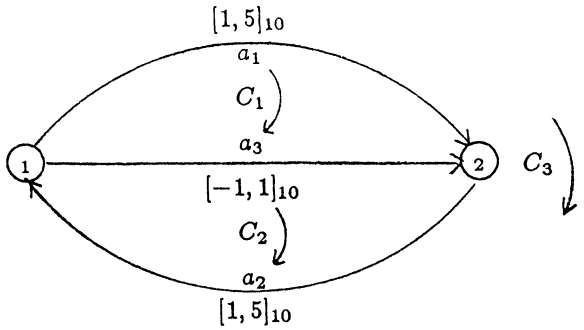

FIG. 2

considerations: in fact, the sufficiency of the above condition would allow a concise certificate, namely a circuit, for the complement of the Hamiltonian circuit problem. Note the difference with respect to the FDP where a similar condition, namely $0 \in D(C)$, is both necessary and sufficient (see [23 p. 193]).

2.3. An algorithm for PESP. In the following we shall borrow the terminology from [23, Chap. 6] and some basic techniques used by the algorithm. We shall also use the following notation: let $P(J, z_K)$ denote a new problem obtained from the network representation of PESP by restricting the span constraints to a subset $J \subset A$, and by fixing the values $z(a)$ for the spans relative to the arcs a of a subset $K \subset J$ to an array z_K of integers. By this notation PESP $= P(A, z_\varnothing)$.

The algorithm for PESP is outlined in Fig. 3. Below we describe its features in detail. An instance of PESP is displayed in Fig. 4.1 and the description of the algorithm will also refer to Fig. 4 for ease of presentation.

First let us note that it is immediate to find a solution for PESP if the network is a tree: it is just a matter of assigning an arbitrary potential to an arbitrary node and then assigning recursively the potentials to the other nodes so that the tensions on the branches of the tree are feasible (see Fig. 4.2).

Indeed, the algorithm is based on the idea of first solving a problem $P(B, z_\varnothing)$, with B the arc set of a spanning tree (N, B) of G and then trying to make the span constraints relative to the chords of the tree feasible one by one. In order to accomplish the latter task, we shall design an implicit enumeration procedure for the integers z relative to the spans of the chords.

It is important to note that the integers z relative to the span constraints of B can be fixed to any value without altering feasibility or infeasibility of the PESP instance at hand, or, more formally, $P(A, z_\varnothing)$ is feasible if and only if $P(A, z_B)$ is feasible for any fixed z_B. In fact, suppose that there exists a feasible potential and that the integer $z(\bar{a})$, with $\bar{a} \in B$, has the feasible value z_1; let (N^+, B^+) and (N^-, B^-) be the trees obtained from (N, B) by removing the arc $\bar{a}$, with $\varepsilon^+(\bar{a}) \in N^+$ and $\varepsilon^-(\bar{a}) \in N^-$. If we replace z_1 with z_2 a new feasible potential is immediately obtained by adding $(z_2 - z_1)T$ to all potentials on N^+.

We call *tree integrality* the property for a given spanning tree (N, B) to admit arbitrary fixed values for the integers $z(a)$, for all $a \in B$ without affecting the feasibility of the PESP instance at hand. It is rather obvious that tree integrality holds for any spanning tree of a PESP instance and so this definition might seem superfluous. However, we shall extend the PESP model in § 2.5 and in this extended model tree integrality will not always hold.

The initialization of the algorithm consists in finding a minimal spanning tree (N, B) with respect to the costs $(d^+(a) - d^-(a))$, sorting the chords as $c_1, c_2, \cdots, c_q$ in order of increasing costs $(d^+(c_i) - d^-(c_i))$ and assigning an initial potential u so that $v(a) = (d^+(a) + d^-(a))/2$, for all $a \in B$. By this choice we have implicitly set $z(a) = 0$, for all $a \in B$. We shall explain in § 2.4 why B has to be a *minimal* spanning tree, the chords have to be *sorted* and the tension is initially set in the *middle* of the span.

The main body of the algorithm is made up of the implicit enumeration procedure to make the chord span constraints feasible one by one. With every PESP instance, a search tree is associated with at most q levels, corresponding to the q chords (obviously $q = |A| - |N| + 1$ is also the number of independent circuits of G). The root of the search tree at level zero corresponds to problem $P(B, z_B)$. All other nodes correspond to some problem $P(J, z_J)$ with $B \subset J \subset A$; at the ith level $J = B \cup c_1 \cup \cdots \cup c_i$. In order to explain the way the successors of a given node are generated in the next level of the search tree we shall consider in detail how the successors of the root node are generated by referring explicitly to Figs. 4.1–4.7.

The successors of Problem $P(B, z_B)$, i.e., the root node, are the Problems $P(B \cup c_1, z_{B\cup c_1})$, differing from each other for the value assigned to $z(c_1)$. The set of admissible values for $z(c_1)$ is generated in the following way (see Fig. 4.3): the tension $v(c_1)$ induced by the feasible potential for Problem $P(B, z_B)$ is equal to 0.5 and it is not feasible for Problem $P(B \cup c_1, z_B)$ since $v(c_1) \notin D(c_1, z)$, for all z.

In order to make Problem $P(B \cup c_1, z_B)$ feasible we have two possibilities, either to raise the tension by at least the quantity $v^+ = 2.5$ so that it belongs to the span at right with value $z(c_1) = 0$ or to lower it by at least the quantity $v^- = 4.5$ so that $v(c_1) \in D(c_1, -1)$. The problem of raising (lowering) the tension on c_1 is the dual problem of a shortest path problem from $\varepsilon^-(c_1)$ to $\varepsilon^+(c_1)$ (from $\varepsilon^+(c_1)$ to $\varepsilon^-(c_1)$) on the network (N, B) with respect to the spans $[d^-(a) + z_a T - v(a), d^+(a) + z_a T - v(a)]$ for $a \in B$ (again see [23]), or alternatively, the dual problem of a shortest path problem on a directed graph (N, B') with B' obtained from B by replacing each arc $a \in B$ with a pair of directed arcs $\varepsilon^-(a) \to \varepsilon^+(a)$ and $\varepsilon^+(a) \to \varepsilon^-(a)$ with lengths $d^+(a) + z_a T - v(a)$ and $v(a) - d^-(a) - z_a T$, respectively. Roughly speaking, these lengths represent the maximum amounts that the tension on the arc a can be "stretched" or "shrunk," respectively. The directed graph (N, B') is shown in Fig. 4.4 (the lengths of each pair of arcs are equal because of the initial choice of the tension in the middle of the spans; this situation does not happen, in general of course, on the other nodes of the search tree).

The tension can be raised up to the span $D(c_1, 0)$ if and only if the shortest path value from $\varepsilon^-(c_1)$ to $\varepsilon^+(c_1)$ is at least v^+ and the tension can be lowered up to the span $D(c_1, -1)$ if and only if the shortest path value from $\varepsilon^+(c_1)$ to $\varepsilon^-(c_1)$ is at least v^-. If so, the new feasible potential is computed in the following way (let us illustrate it for the span $D(c_1, 0)$, the other case is symmetric): Dijkstra's algorithm generates a tree of shortest paths from $\varepsilon^-(c_1)$ to all other nodes by appending recursively the nearest node to an expanding tree starting from the source $\varepsilon^-(c_1)$. Let $w(\varepsilon)$ be the shortest path value from $\varepsilon^-(c_1)$ to a generic node ε. Then, in order to compute the new feasible potential it is not necessary to run Dijkstra's algorithm to its full completion; it is enough to stop it as soon as a node $\bar{\varepsilon}$ is reached with distance $w(\bar{\varepsilon}) \geqq v^+$. Let N' be the set of nodes reached by Dijkstra's algorithm up to this point with the exclusion of the last node $\bar{\varepsilon}$. Then the potential is updated according to

$$u(\varepsilon) := \begin{cases} u(\varepsilon) + w(\varepsilon) - v^+ & \text{if } \varepsilon \in N' \\ u(\varepsilon) & \text{otherwise.} \end{cases}$$

In the example of Figs. 4.1–4.7, both raising and lowering the tension of the stated amounts are possible, so both $z(c_1) = -1$ and $z(c_1) = 0$ correspond to admissible successors of $P(B, z_B)$. Other admissible successors can be generated by raising (lowering) the tension even more up to the span next to the right (left), i.e., $D(c_1, 1)$ ($D(c_1, -2)$). In this case it is required that the shortest path value be at least 12.5 (14.5). For this example no feasible tension can be established for the spans $D(c_1, 1)$ and $D(c_1, -2)$ and this implies in turn that no feasible tension can exist for all spans $D(c_1, z)$ with $z \geqq 1$ or $z \leqq -2$.

So the successors of $P(B, z_B)$ are all the problems $P(B \cup c_1, z_{B\cup c_1})$ with values $z(c_1)$ giving raise to feasible potentials (these $z(c_1)$ constitute a set of "adjacent" integers), plus two infeasible problems corresponding to the integers immediately at the left and at the right of the set of feasible integers. These last two problems must belong to the search tree because they are necessary for the algorithm to backtrack.

The above discussion for the successors of the root node of the search tree is also valid for the successors of any node $P(J, z_J)$. So the structure of the search tree is the following: each feasible $P(J, z_J)$ has at least two successor problems and exactly two of them are nonfeasible. Nonfeasible problems do not have successors. Feasible solutions

of the PESP problem are found as the feasible potentials for the problems at the qth level (the bottom) of the search tree. See in Fig. 4.5 the search tree of the example. Note that a infeasible instance of PESP may have less than q levels.

It is important to note that whenever the tension $v(c)$ on a certain chord c cannot be made feasible with respect to a certain span $D(c, z)$, the shortest path between the extremes of the chord together with the chord itself form a circuit for which the span constraints are too tight to accommodate a feasible potential. Let us call this circuit a *blocking circuit*, and the other chords besides c possibly belonging to the blocking circuit *blocking chords*. We shall use the phrase *blocking gap* to indicate the quantity by which $v(c)$ differs from the span to which it is modified (i.e., the blocking gap is either $d^-(c) + z_i T - \max v(c)$ or $\min v(c) - d^+(c) - z_i T$). As we shall see, the information provided by a blocking circuit is very valuable.

The algorithm we suggest performs a depth-first visit of the search tree by selecting at each level the most central successor in the next level, and backtracking to the most central nonvisited node of the previous level whenever the two infeasible nodes of the level are visited. See in Fig. 4.6 the nodes of the search tree actually visited. Note that, in view of the previous considerations, a visit of a node consists of a run of the Dijkstra's algorithm.

The performance of this "naive" algorithm can be improved if we introduce a device, which may be called *accelerated backtracking*, based on the following considerations: when the algorithm backtracks from a certain level, this is because a certain set of blocking circuits has been discovered at deeper levels. In order to find a feasible potential the only integers z of the previous levels which should be changed are the ones relative to the blocking chords. Therefore there is no need to guess other z values at the level immediately above if this refers to a chord nonpresent in any blocking circuit. Thus the accelerated backtracking device consists in backtracking until a level is found corresponding to a blocking chord.

Figure 4.7 shows the nodes of the search tree of the example actually visited by using the accelerated backtracking.

In order to appreciate the importance of the accelerated backtracking let us consider a PESP instance whose network consists of two networks G_1 and G_2 linked through a node. The instance is actually made up of two independent instances, but let us suppose that we have not realized the existence of this simpler structure and solve the problem globally with the "naive" algorithm. Let us also suppose that the problem is infeasible on G_2 and feasible on G_1 and that the chords are sorted so that all chords in G_1 correspond to the first levels of the search tree. So the algorithm will first find a feasible potential with respect to G_1 and then will discover that a feasible potential for the whole network cannot be found with the fixed z values in G_1. Therefore the "naive" algorithm will backtrack to guess other z values for the chords in G_1. But this is actually a waste of time since these values do not affect feasibility at all on G_2. This fact can be discovered by looking at the blocking chords. In fact all blocking circuits in G_2 are necessarily confined in G_2, so when backtracking from G_2 to G_1, the algorithm may realize that there are no blocking chords in G_1, thus making a unique backtracking up to the root node and concluding infeasibility of the instance.

The algorithm can also be embedded with a routine which records some of the blocking circuits found as well as the relative gaps. This kind of information turns out to be useful, in case of infeasibility, if the possibility is considered of relaxing some span constraints in order to find a feasible solution.

2.4. Considerations for the probabilistic analysis of the algorithm. With regard to the worst-case computational complexity of the algorithm described in the previous section, we may note that it is given by $O(s(n)n^2)$, with $O(n^2)$ being the complexity relative

to the visit of a node (running the Dijkstra's algorithm) and $s(n)$ being a bound on the number of nodes of the search tree. Obviously $s(n)$ is exponential in the worst case. Nevertheless it is possible to arrange things so that at least the probabilistic behaviour is acceptable.

Of course the less the nodes of the search tree are the faster the computation is. It turns out that the number of nodes not only depends on the instance data, but also on the way the data are "fed" to the algorithm. Intuitively, it is better to have the most "prolific" nodes at the bottom rather than at the top of the search tree, and this situation can be achieved if the arcs considered first are the ones with narrowest spans. This explains why in the stated algorithm we choose B as a minimal spanning tree and the chords are sorted. Moreover, the initial value for the tensions is chosen in the middle of the span in the presumption that selecting the most central successor of a given node of the search tree gives the highest probability of finding a feasible successor.

These considerations are rather informal. However, it is possible to support them with more rigorous arguments. In fact, each feasibility problem $P(J, z_J)$ is a relaxation of $P(A, z_J)$ and we may think of it as an optimization problem with objective function $\sum_{a \in J} \text{distance}\,(v(a), D(a, z_a))$. Therefore a node of the search tree is nonfeasible if and only if the optimal value of its corresponding relaxation is strictly positive. Averaging on a random set of instances the probability for a node to be nonfeasible is monotonically decreasing with respect to the span lengths of the arcs in J and increasing with respect to $|J|$. The same conclusion can be reached if we consider that nonfeasibility of a node means that a shortest path between two given nodes is shorter than the quantity by which $v(a)$ "misses" the real span $D(a, z_a)$.

Now, according to Stone and Sipala [27], if the probability of being forced to a backtrack in a binary search tree (with consequent cutoff) is constant at any level, then an algorithm will visit the tree probabilistically in linear time with respect to the tree depth. Unfortunately this is a limiting case because any practical algorithm behaves in such a way that the cutoff probability increases, due to the accumulation of information, as the visit goes deeper. What can be done about this is to keep this increase as small as possible. We may apply these conclusions to our search tree since they are valid even if a nonbinary search tree is concerned. Therefore in view of the previous observation on the probabilities for a node to be nonfeasible in the PESP algorithm, the best thing to do is start with a minimal spanning tree B and then consider sorted chords.

Tables 1, 2, and 3 report some experimental computations for the PESP algorithm of Fig. 3. The data of each instance have been randomly generated with the following rules: first the directed graph corresponding to the network PESP model has been generated with independent arc probabilities given by $d/(n-1)$, for all arcs (i, j) with $i < j$, where d is the desired average degree at each node. Some additional arcs have been possibly generated to make the network connected. Then the span constraints have been generated for all these arcs as $[\lfloor aT \rfloor, \lfloor (a+b)T \rfloor]$ where a and b are two independent random numbers uniformly generated between 0 and 1 and the period T has been set to 100. The rounding is necessary to avoid round-off errors and consequent anomalous behaviour of some routines.

The values of d have been set to: 1.50, 2.00, 2.50, 3.00, 3.50, 4.00, 4.50, 5.00. These values are the most significant ones, since for smaller values the algorithm runs almost without backtracking to a feasible solution, whereas for larger values the span constraints are so binding to make the depth of the search tree very small, and infeasibility is reported almost immediately. The values of n have been set to: 25, 50, 75, 100, 150, 200. Twenty instances have been generated for each fixed value of d and n.

Table 1 reports the average cpu time in milliseconds (on a VAX 11/780). The smaller figures appearing as subscripts and superscripts of the average times refer to the

smallest and to the largest cpu time, respectively, over the twenty instances. A threshold value of 100 seconds cpu time was set to interrupt the execution of the algorithm. The number of instances out of twenty requiring more than 100 seconds and thus being interrupted is reported within brackets in the relative entry of the table if greater than zero. In case of interruption, the cpu time was set to 100 seconds and the instance considered infeasible.

Table 2 reports the average, the smallest, and the largest number of visited nodes of the search tree, and Table 3 reports the percentage of feasible instances over the same twenty instances.

The figures of Table 2 show a nondramatic increase in the number of visited nodes as the instance size increases, thereby confirming the previous theoretical considerations. The cpu times increase at a higher rate because on each node a shortest path problem has to be solved on a network whose size increases with the search tree level. However, average figures are poorly indicative of the behaviour of the algorithm: as is typical of NP-complete problems, the running times for homogeneous instances range over a wide spectrum of values. Whereas for most instances the running times are acceptable, once in a while an instance appears with a very long running time. Due to these reasons we also report the smallest and the largest running times to give a more precise idea of the behaviour of the algorithm.

It is fair to say that for instances with large spans, say almost as large as the period, the algorithm does not behave as satisfactory as for uniformly random spans. This is the case for instance of the Hamiltonian circuit problem solved through PESP (see Thm. 1). Moreover there are some scheduling problems where such large spans arise naturally as we shall see in §§ 2.7 and 3.4.

2.5. The extended PESP (EPESP). As we shall see in § 3, the presence of resources may complicate the periodic character of the scheduling problem by introducing different periods mT (for integers m), which have to be taken into consideration simultaneously (see also for instance [6], [14], [21]). For instance, spans of the type $[d^-, d^+]_{mT}$ may be present in the model.

In principle, the PESP model can handle these situations as well since it is possible to consider the larger period $\hat{m}T$, with $\hat{m}$ the least common multiple of the integers m and to transform each span $[d^-, d^+]_{mT}$ into

$$\bigcup_{k=1 \cdots \hat{m}/m} [d^- + kmT,\ d^+ + kmT]_{\hat{m}T}$$

and to express in turn this union as an intersection of spans, as explained earlier.

However, this is unnecessarily expensive from a computational point of view because each of these span constraints produces several arcs, thus increasing the number of chords and expanding the search tree.

It is more convenient to refer directly to the network model of PESP and then to adapt with minor changes the previous algorithm to this case. More precisely we may define the Extended Periodic Event Scheduling Problem (EPESP) in the following way:

Given

- a period T;
- a network (N, A);
- positive integers $m(a)$, $\forall a \in A$;
- real spans $D(a, z)$, $\forall a \in A$, $\forall z \in Z$ where $D(a, z) = [d^-(a) + zm(a)T,\ d^+(a) + zm(a)T]$ also denoted as $D(a, z) = [d^-(a), d^+(a)]_{m(a)T}$;

find a potential u such that $v(a) \in D(a, z(a))$ for some $z(a) \in Z$, for all $a \in A$. Then $\tau(\varepsilon) = u(\varepsilon) \bmod T$.

Note the difference with PESP where $m(a) = 1$, for all a.

The practical utility of this model will be clear later when activities and resources will be taken into account. For the moment let us focus on a solution method for EPESP.

First let us note that the PESP algorithm is not adequate to solve EPESP since tree integrality does not necessarily hold for all spanning trees of the network (N, A). In fact, if the instance is feasible, and $z(a) = z_1$ is a feasible integer value for the branch a of a spanning tree (N, B), and we replace z_1 with z_2, then the tension on an arc c belonging to the cutset generated by a changes by $(z_2 - z_1)m(a)T$. Since it is required that $d^-(c) \leqq v(c) - z(c)m(c)T \leqq d^+(c)$, the change in the tension $v(c)$ will not affect feasibility of the corresponding span constraint if $(z_2 - z_1)m(a)$ is an integer multiple of $m(c)$. So if we want to assign any integer value z to the branch a we must require that $m(c)$ divides $m(a)$.

Consider for instance the following example: $N = \{\varepsilon_1, \varepsilon_2\}$, $A = \{a_1, a_2\}$, $\varepsilon^-(a_1) = \varepsilon^-(a_2) = \varepsilon_1$, $\varepsilon^+(a_1) = \varepsilon^+(a_2) = \varepsilon_2$, $D(a_1) = [1, 5]_{10}$, $D(a_2) = [7, 8]_5$. Then if $B = \{a_2\}$ the algorithm will report infeasibility if $z(a_2)$ is fixed to an odd value and feasibility if $z(a_2)$ is fixed to an even value, whereas if $B = \{a_1\}$ the algorithm will report feasibility for any fixed value of $z(a_2)$.

In general terms, let us call *chord integrality* the condition for a given chord c of a spanning tree that $m(c)$ divides all the integers $\{m(a)\}_{a \in C(c)}$, with $C(c)$ the circuit generated by c, and let us call *branch integrality* the condition for a given branch a of a spanning tree that all $m(c)$ in the cutset generated by a divide $m(a)$; then tree integrality holds for the given spanning tree if branch integrality is verified by all branches or, equivalently, chord integrality is verified by all chords. See Fig. 6.4 for an example of a spanning tree for which tree integrality holds.

Thus if a spanning tree satisfying tree integrality is available, the PESP algorithm can be used without modifications. If there is no spanning tree satisfying tree integrality (or if it is computationally hard to find one such tree), it is nevertheless possible to modify the PESP algorithm in the following way.

Let the root node of the search tree correspond to the problem $P(B', z_{B'})$ where $B' \subset B$ is a set of branches of a spanning tree B for which branch integrality holds (B' can be void). Then let the algorithm run as before with $\{c_1, \cdots, c_{q'}\} = B \backslash B'$ and $\{c_{q'+1}, \cdots, c_q\} = A \backslash B$. The only difference with the PESP algorithm consists in the generation of successor nodes at the first q' levels. More precisely, when $c_i \in B \backslash B'$ the successor nodes of $P(J, z_J)$, with $J = B' \cup c_1 \cup \cdots \cup c_{i-1}$ correspond to the problems $P(J \cup c_i, z_{J \cup c_i})$ with $z_{c_i} = 1, \cdots, \hat{m}/m(c_i)$ and $\hat{m}$ the least common multiple of the $m(a)$ relative to the arcs a belonging to the cutset generated by c_i (including the arc c_i).

Note that $\hat{m}/m(c_i) = 1$ if branch integrality holds for c_i. So this modification is actually a generalization of the PESP algorithm. We have still to investigate whether it is more convenient from the computational point of view to find a set B' of maximal cardinality, thus reducing the number of levels, or to find a set B' with spans as narrow as possible, thus reducing the number of successors.

2.6. Periodic events sequencing. A set of periodic events cannot be ordered with respect to their schedules since these are elements of the group R/T. However a weaker concept of ordering is preserved if we consider how the occurrences of m periodic events are displaced in a (real) time interval of length T. More formally, we say that m real numbers $a_1, \cdots, a_m$ are *sequenced with respect to* T if

$$(a_i + b) \bmod T \leqq (a_{i+1} + b) \bmod T$$

holds true for some real b and for all $i = 1, \cdots, m - 1$ and we say that they are *strictly sequenced* if strict inequality holds in the above relationship.

Then we shall say that *m periodic events* $\varepsilon_1, \cdots, \varepsilon_m$ *are sequenced* or *are scheduled in sequence* (*strictly sequenced* or *scheduled in strict sequence*) if the real numbers $t(\varepsilon_1), \cdots, t(\varepsilon_m)$ are sequenced (strictly sequenced). Of course two scheduled periodic events are always sequenced and for m scheduled periodic events exactly m permutations of them correspond to strictly sequenced periodic events.

Constraints requiring certain subsets of periodic events to be sequenced in a definite way will appear naturally when we introduce resources in the model.

It is not difficult to impose the condition that m periodic events have to be scheduled in sequence. The following lemma provides a key property.

LEMMA. *Let* $a_1, \cdots, a_m$ *be real numbers such that the numbers* a_i mod T *are not all equal, and let* z_{ij} *be integers so that* $0 \leqq a_j - a_i - z_{ij}T < T$. *Then the* TSP *with costs* $(-z_{ij})$ *has optimal value* 1 *and the optimal solution is exactly the permutation which sequences the* a_i*'s.*

Proof. Let z_i be integers so that $0 \leqq a_i - z_iT < T$. Let $\alpha_i = a_i - z_iT$. Therefore we have

$$0 \leqq \alpha_j + z_jT - \alpha_i - z_iT - z_{ij}T < T \quad \text{i.e.,} \quad 0 \leqq \alpha_j - \alpha_i - (z_{ij} - z_j + z_i)T < T.$$

Note now that the TSP with costs $-(z_{ij} - z_j + z_i)$ is equivalent to the TSP with costs $(-z_{ij})$. However it is obvious that $-(z_{ij} - z_j + z_i)$ must be either 1, if $\alpha_j < \alpha_i$, or 0, if $\alpha_j \geqq \alpha_i$ and therefore, since the α_i are not all equal, the optimal solution is the one ordering the α_i's with optimal value 1. □

With the aid of the previous lemma, it is possible to handle a sequence or strict sequence condition by means of span constraints. Actually if the periodic events $\varepsilon_1, \cdots, \varepsilon_m$ have to be scheduled in sequence with respect to T (besides other possible span constraints originally present in the problem), m arcs

$$s_1 := (\varepsilon_1, \varepsilon_2), \cdots, s_m := (\varepsilon_m, \varepsilon_1)$$

forming a circuit S have to be added to the network with spans

$$D(s_i, z) := [zT, T + zT].$$

If $\varepsilon_1, \cdots, \varepsilon_m$ have to be strictly sequenced, we have to add the following spans:

$$D(s_i, z) := [zT, T - U + zT]$$

where U is the greatest commensurability unit of the instance data (if we suppose the data are rational numbers).

Then for any potential $u(\varepsilon_i)$ we compute z_i: $v(s_i) \in D(s_i, z_i)$ (the tie occurring if $u(\varepsilon_i) - u(\varepsilon_j)$ mod $T = 0$ is broken by choosing the smaller z). From the lemma we know that the events $\varepsilon_1, \cdots, \varepsilon_m$ are scheduled in sequence if and only if $\sum_i z_i = -1$, unless $\Pi \cdot u(\varepsilon_i)$ are not all equal, in which case the events are trivially sequenced.

Therefore the algorithm runs as for a usual PESP with the only difference being that whenever $s_i \in S$ and $S \subset J \cup s_i$, i.e., the last arc of S is appended to J, then the value $z(s_i)$, according to the previous results, has to be fixed to the value

$$z(s_i) = -1 - \sum_{j \neq i} z_{s_j}.$$

Apparently a sequence condition on m periodic events introduces $m - 1$ new levels to the search tree (not counting the one with fixed z). Sometimes this added complexity can be reduced if the span $D(s_i)$ contains a previous span D for the same pair of periodic events, in which case D can freely subsume $D(s_i)$, as it happens for instance, in the problems dealt with in § 3.3.

For the tie to be broken for the choice of $z(s_i)$ mentioned above, it can be seen that the PESP algorithm itself will break the tie automatically as required and so there is no need to explicitly impose this extra condition.

2.7. PESP and ΔTSP. The results of the previous section and the reduction given by Theorem 1 show that there is a close connection between PESP and ΔTSP. In fact it is not difficult to see that for any instance of the symmetric ΔTSP with costs c_{ij}, the question "does there exist a tour with cost less than or equal to T?" is equivalent to the question about feasibility of the PESP instance with period T and spans $[c_{ij}, T - c_{ij}]$ for all pairs of periodic events arbitrarily ordered. Incidentally this is an alternative proof of NP-completeness for PESP.

By this transformation a ΔTSP instance can be obviously solved via the PESP algorithm described in §§ 2.3 and 2.4. However, due to the particular spans involved in this case, that algorithm does not perform in a satisfactory way. We must point out that similar spans can occur for periodic scheduling problems (see § 3.4). Therefore, whenever this situation is faced, it seems to be advisable to solve the scheduling problem by using known techniques developed for ΔTSP.

THE PESP ALGORITHM.

```
compute minimal spanning tree B;
sort chords c_1, ..., c_q;
compute potentials on B;
                              {initialization search tree visit}
J := B; i := 1; status[1] := new; for i := 1 to q do blocking_chord[i] := false; feasible := false; backtracking := false;
                              {search tree visit}
 {z_i is the next integer to be guessed at level i. Each level can be in one of three states: new, all times when it is reached
  from above; the status new is immediately reset to alternating to guess integer values alternating away from the most
  central integer; when one of the two extreme infeasible nodes is visited the status is reset to one_way and integer
  values are looked for in one direction only.
  Shortest_path(ε1,ε2,J,v̄,feasible) is a routine computing the shortest path from ε1 to ε2 on the network J via the Dijkstra's
  algorithm with an anticipated stop if a node is found whose distance from ε1 exceeds v̄; feasible is set false if and only
  if the distance from ε1 to ε2 is shorter than v̄. }
while 1 ≦ i ≦ q do
begin
  repeat              {trying to find a feasible solution at level i}
    if status[i] = new
    then begin        {executed only if level i is reached from level i − 1}
      status[i] := alternating;
      if ∃z̄: v(c_i) ∈ D(c_i, z̄)
      then begin      {tension is already feasible}
        z_i := z̄ + 1;
        feasible := true;
        step[i] := 1;
        direction[i] := 1;
      end
      else begin      {tension is not feasible}
        compute z^1 := argmin_z{d^-(c_i) + zT − v(c_i) ≧ 0};
        compute z^2 := argmin_z{v(c_i) − d^+(c_i) − zT ≧ 0};
        if d^-(c_i) + z^1T − v(c_i) ≦ v(c_i) − d^+(c_i) − z^2T
        then begin {tension should be raised}
          z_i := z^1;
          direction[i] := 1;
        end
        else begin {tension should be lowered}
          z_i := z^2;
          direction[i] := −1;
        end;
        step[i] := 0;
      end;
    end
```

FIG. 3

```
    else begin      {either remaining in level i or reaching level i from below}
      if direction[i] = 1
      then shortest_path(ε⁻(c_i),ε⁺(c_i),J,d⁻(c_i) + z_iT − v(c_i), feasible)
      else shortest_path(ε⁺(c_i),ε⁻(c_i),J,v(c_i) − d⁺(c_i) − z_iT, feasible);
      if feasible
      then begin
        update_potential;
        if status[i] = alternating
        then begin
          step[i] := step + 1;
          direction[i] := −direction[i];
        end;
        z_i := z_i + direction[i]·step[i];
      end
      else begin
        for j := 1 to i − 1 do
          if c_j ∈ blocking_circuit then blocking_chord[j] := true;
        case status[i] of
          alternating: begin
                         direction[i] := −direction[i];
                         z_i := z_i + direction[i]·step[i];
                         step[i] := 1;
                         status[i] := one_way;
                       end;
          one_way:  backtracking := true;
        end;
      end;
    end;
  until feasible or backtracking;
  if feasible       {going down one level}
  then begin
    J := J ∪ c_i;
    i := i + 1;
    status[i] := new;
    feasible := false;
  end;
  if backtracking    {accelerated backtracking}
  then begin
    repeat
      i := i − 1
      J := J\c_i;
    until (i = 1) or blocking_chord[i];
    if blocking_chord[i]
    then for j := 1 to i do blocking_chord[j] := false
    else i := i − 1;
    backtracking := false;
  end;
end;
if i = 0 then output ("PESP is infeasible");
if i := q + 1 then output solution;
```

FIG. 3 (*Continued*)

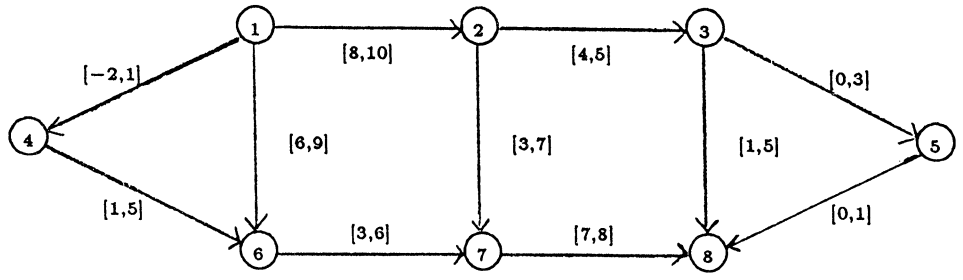

FIG. 4.1. *A* PESP *instance*.

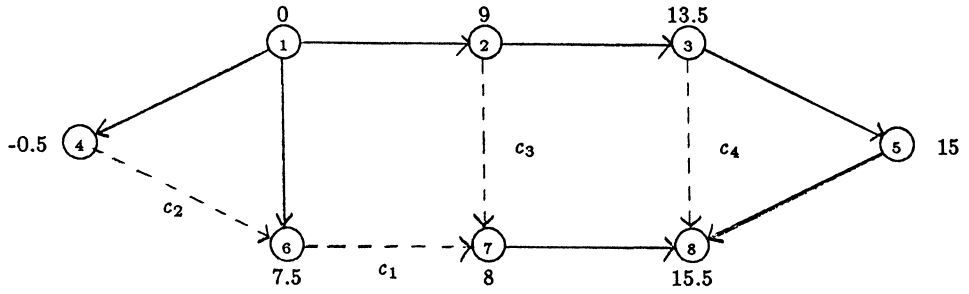

FIG. 4.2. *Spanning tree and initial potentials.*

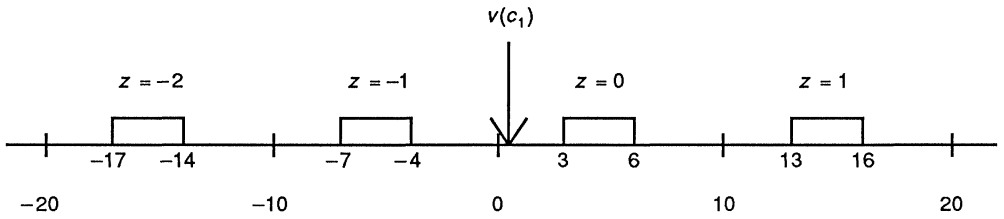

FIG. 4.3. *Admissible $z(c_1)$ values.*

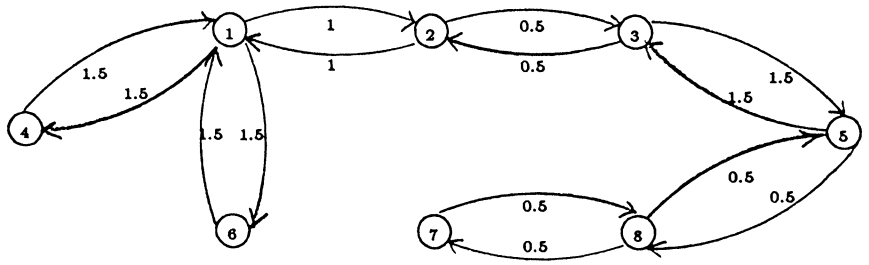

FIG. 4.4. *Graph (N, B').*

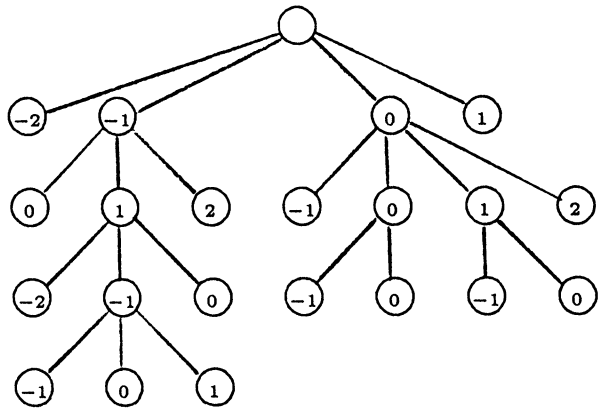

FIG. 4.5. *Full search tree.*

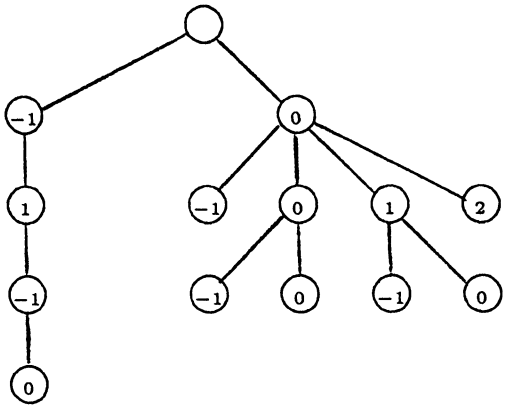

FIG. 4.6. *Nodes visited by naive algorithm.*

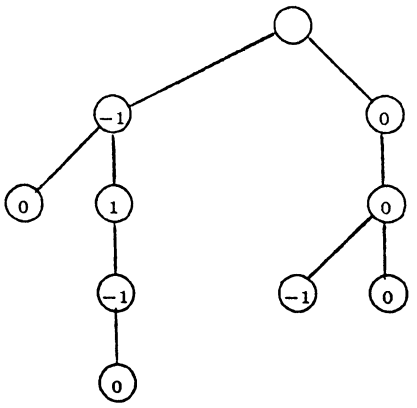

FIG. 4.7. *Nodes visited with accelerated backtracking.*

TABLE 1
Cpu *times in milliseconds.*

	25	50	75	100	150	200
1.50	59^{120}_{40}	153^{210}_{120}	303^{390}_{240}	522^{940}_{410}	1066^{1170}_{950}	1817^{2150}_{1570}
2.00	76^{270}_{50}	184^{220}_{150}	384^{590}_{310}	671^{1080}_{520}	1923^{12290}_{1030}	2274^{5120}_{1670}
2.50	98^{240}_{60}	265^{830}_{160}	432^{550}_{340}	1098^{7800}_{560}	3522^{31970}_{1140}	2995^{22690}_{1530}
3.00	124^{250}_{50}	334^{1010}_{190}	1534^{10830}_{370}	2247^{16330}_{560}	5680^{40240}_{1220}	17273^{100000}_{2370} (2)
3.50	156^{470}_{60}	538^{3290}_{190}	1792^{17880}_{390}	5953^{49400}_{770}	8557^{42850}_{1440}	45293^{100000}_{2600} (8)
4.00	169^{440}_{70}	740^{6270}_{240}	1577^{7620}_{470}	7930^{99000}_{770}	17748^{100000}_{1820} (1)	34864^{100000}_{2880} (5)
4.50	171^{520}_{9}	969^{7030}_{270}	1172^{5550}_{510}	1986^{19840}_{830}	26393^{100000}_{1800} (4)	5578^{20230}_{3320}
5.00	186^{770}_{100}	705^{2820}_{310}	2451^{7550}_{620}	2755^{15950}_{960}	15177^{100000}_{2170} (2)	10678^{77740}_{3940}

TABLE 2
Number of visited nodes.

	25	50	75	100	150	200
1.50	8_3^{16}	13_3^{28}	18_3^{28}	26_{10}^{41}	42_{32}^{56}	55_{37}^{74}
2.00	12_3^{25}	19_3^{29}	32_6^{45}	40_3^{61}	59_3^{166}	71_3^{113}
2.50	16_3^{45}	27_3^{60}	38_3^{56}	45_3^{220}	86_3^{651}	86_3^{219}
3.00	16_3^{45}	33_3^{68}	68_3^{449}	73_3^{558}	89_3^{553}	213_3^{1132} (2)
3.50	23_3^{86}	37_3^{269}	51_3^{449}	63_3^{315}	114_3^{535}	306_3^{818} (8)
4.00	20_3^{89}	35_3^{350}	34_3^{189}	159_3^{2039}	142_3^{1050} (1)	194_3^{746} (5)
4.50	14_3^{47}	45_3^{292}	18_3^{114}	17_3^{229}	243_3^{1005} (4)	15_3^{100}
5.00	14_3^{78}	27_3^{140}	45_3^{170}	27_3^{210}	100_3^{857} (2)	32_3^{308}

TABLE 3
Percentage of feasible problems.

	25	50	75	100	150	200
1.50	95	90	85	95	100	100
2.00	80	80	95	90	70	75
2.50	70	60	80	50	55	65
3.00	50	75	55	45	35	35 (2)
3.50	50	40	20	30	35	10 (8)
4.00	25	20	5	10	0 (1)	0 (5)
4.50	15	20	0	0	5 (4)	0
5.00	5	10	0	0	0 (2)	0

3. Activities and resources.

3.1. Basic definitions. A *periodic activity* γ is an ordered pair of periodic events $(\varepsilon^-(\gamma), \varepsilon^+(\gamma))$, corresponding to its beginning and ending. A periodic activity γ is said to be scheduled if both $\varepsilon^-(\gamma)$ and $\varepsilon^+(\gamma)$ are scheduled and a nonnegative real number $x(\gamma)$, i.e., its *duration*, is assigned in such a way that

$$x(\gamma) = t(\varepsilon^+(\gamma)) - t(\varepsilon^-(\gamma)) + z(\gamma)T \geqq 0 \text{ for some } z(\gamma) \in Z.$$

We shall be concerned with a finite set Γ of periodic activities.

The pth occurrence of the periodic activity γ is the activity starting at $t(e^-(\gamma, p))$ and ending at $t(e^-(\gamma, p)) + x(\gamma)$. It is denoted by (γ, p).

It is interesting to deal with activities using resources. By *resources* we mean a finite set $\mathscr{R}$, where each resource $R \in \mathscr{R}$ is in turn a finite set of *resource units* of some definite type. To each periodic activity γ, a set $\mathscr{R}(\gamma) \subset \mathscr{R}$ is associated, which corresponds to the resources needed by the activity γ. Correspondingly, we define $\Gamma(R) = \{\gamma: R \in \mathscr{R}(\gamma)\}$, that is, the set of activities needing the resource R.

The resource units constitute a set of indistinguishable units and so any activity needing a particular resource may use any unit of it. Therefore we have to solve at the same time the problem of scheduling the activities and the problem of assigning the activities to the resource units. Formally we define as a *resource assignment* for the resource R and denote it by RA_R *an assignment of* (γ, p) *to exactly one* $r \in R$ *for all* $R \in \mathscr{R}(\gamma)$ *and all* $p \in Z$. For ease of presentation we shall occasionally simplify the notation by dropping the dependence on the particular resource, if it is not necessary to emphasize this dependence.

A resource assignment RA can be viewed as a two-dimensional array, with rows corresponding to periodic activities and columns corresponding to occurrences, whose (γ, p) entry refers to some resource unit $r = RA(\gamma, p) \in R$. By $RA(p)$ we denote the pth column of RA (see Fig. 5).

For any two periodic activities γ_i and γ_j using the same resource, let $s(\gamma_i, \gamma_j)$ be the least amount of time needed by a resource unit released by activity γ_i to be available for the activity γ_j. This time will be called the *set-up time*.

Depending on how the activities are scheduled, the resource assignment may or may not be feasible, where infeasibility is intended as the simultaneous request in the real time of the same resource unit by two different activities, or also by two different occurrences of the same activity (if for instance the duration of the activity is longer than the period).

DEFINITION. An RA is *feasible* if $RA(\gamma_i, p) = RA(\gamma_j, q)$ and $(\gamma_i, p) \neq (\gamma_j, q)$ imply that either $t(e^-(\gamma_i, p)) + x(\gamma_i) + s(\gamma_i, \gamma_j) \leqq t(e^-(\gamma_j, q))$ or $t(e^-(\gamma_j, q)) + x(\gamma_j) + s(\gamma_j, \gamma_i) \leqq t(e^-(\gamma_i, p))$.

We shall model the problem of finding a feasible schedule (feasible with respect to the span constraints) together with a feasible resource assignment (as defined above) through a feasibility problem with respect to span constraints only, thereby exploiting the results of § 2. In order to achieve this result, we first need to show that the resources are used in a peculiar periodic way. Thereafter we shall show that feasibility of a resource assignment can be expressed via span constraints.

First note that, due to the finiteness of R, there must be two occurrences p and p' such that $RA(p) = RA(p')$. We shall make the following assumption.

Assumption. $RA(p + z) = RA(p' + z)$ for all $z \in Z$.

This way the resource assignment exhibits a periodic behaviour with period $T' = |p' - p| T = mT$. We call this time period the *resource cycle*. Besides this cyclic aspect, a resource assignment is also inherently periodic with respect to T as shown in the following theorem.

THEOREM 3. *For any fixed activity schedule there exists a resource assignment RA, minimizing the number of used resource units, such that $RA(p + 1)$ is obtained from $RA(p)$ by applying a fixed permutation PR (the same for all p's) to the elements of R.*

Hence a resource assignment is determined by the following two unidimensional arrays of lengths $|\Gamma|$ and $|R|$, respectively:

$T = 10$
$\mathscr{R} = \{R\}$, $R = \{r_1, r_2, r_3, r_4, r_5\}$, $\Gamma = \{\gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5, \gamma_6\}$
$\Gamma(R) = \Gamma$, $\mathscr{R}(\gamma_i) = \mathscr{R}$ $\forall i$, $s(\gamma_i, \gamma_j) = 0$

Activity schedule

	$t(\varepsilon^-)$	$t(\varepsilon^+)$	$x(\gamma)$	$z(\gamma)$
γ_1	0	5	5	0
γ_2	7	2	5	1
γ_3	2	3	1	0
γ_4	4	8	4	0
γ_5	3	6	13	1
γ_6	7	1	4	1

$RA(0) = [r_1, r_1, r_2, r_2, r_4, r_5]$

Resource assignment

p	0	1	2	3	4	5	6
γ_1	r_1	r_2	r_3	r_1	r_2	r_3	r_1
γ_2	r_1	r_2	r_3	r_1	r_2	r_3	r_1
γ_3	r_2	r_3	r_1	r_2	r_3	r_1	r_2
γ_4	r_2	r_3	r_1	r_2	r_3	r_1	r_2
γ_5	r_4	r_5	r_4	r_5	r_4	r_5	r_4
γ_6	r_5	r_4	r_5	r_4	r_5	r_4	r_5

$PR = (2, 3, 1, 5, 4) = (1, 2, 3)(4, 5)$

Resource plan

$Q = \{\{\gamma_1, \gamma_2, \gamma_3, \gamma_4\}\{\gamma_5, \gamma_6\}\}$ $M = (3, 2)$ $PA = (2, 3, 4, 1, 6, 5) = (1, 2, 3, 4)(5, 6)$

FIG. 5. *Example of resource assignment and resource plan.*

1) a particular column $RA(p)$ of the resource assignment;

2) a permutation PR on R.

Proof. We limit ourselves to outlining the proof; the reader will have no difficulties in filling in the necessary details. First we need to define a *periodic bipartite graph*. We say that (V, U, E) is a periodic bipartite graph if $|V| = |U| = n = mk$ with m and k positive integers such that

$$(v_i, u_j) \in E \Leftrightarrow (v_{(i+k) \bmod n}, u_{(j+k) \bmod n}) \in E.$$

A *weighted* periodic bipartite graph has weights obeying a similar relationship.

We build a complete weighted periodic bipartite graph to represent feasible resource assignments with resource cycle mT and $k = |\Gamma(R)|$. In particular $v_{jk+i} \in V$ corresponds to the event related to the ending of the jth occurrence (within the resource cycle) of the ith activity of $\Gamma(R)$. Similarly, vertices of U are related to beginnings of activity occurrences. Two occurrences of the same activity distant m periods are identified, according to the previous assumption.

Recall that all activities have been scheduled. Assigning a resource unit, whose use by the jth occurrence of the ith activity has just terminated, to the pth occurrence of the qth activity, is equivalent to matching the vertices v_{jk+i} and u_{pk+q} of G. The weight to be attached to this arc is simply given by the actual idle time the resource unit would spend during this transfer. The weights have an obvious periodic character.

The total weight of any matching plus the activity durations is equal to a certain multiple of the resource cycle. This multiple is also the number of resource units necessary to produce a feasible resource assignment according to the given matching.

The given resource assignment (with resource cycle mT) corresponds to a definite matching on the periodic graph just defined. Due to the previous observation, finding a matching of minimum weight is equivalent to finding a possibly different resource assignment with less or equal resource units. We are therefore interested in finding a matching of minimum weight in a weighted complete periodic bipartite graph.

Our claim is that there exists an optimal matching which is a periodic subgraph of (V, U, E) (with the same m and k). We recall that a weighted bipartite matching can be solved through a primal-dual method by solving a certain number of cardinality bipartite matching problems (see [20]).

Our subclaim is that there exists a maximum cardinality matching for a periodic bipartite graph which is periodic as well. Let us suppose that a periodic matching is given; then if it is not maximum there exist augmenting paths. Since the matching is periodic these paths are periodic as well. The idea is to augment simultaneously all these paths in order to have another periodic matching of larger cardinality. However this is possible only if the paths do not intersect. So we want to prove that there always exists a periodic set of nonintersecting augmenting paths. Therefore let us suppose that the augmenting path $v_i \rightarrow u_j$ intersects the augmenting path $v_{i+k} \rightarrow u_{j+k}$ with v_{i+k} nearer to the intersection than v_i. Then we may "exchange" the paths in the intersection to form a new augmenting path $v_{i+k} \rightarrow u_j$. By repetitively doing so for all augmenting paths, we are left with a new periodic set of augmenting paths plus an alternating circuit intersecting all augmenting paths. This circuit can be dropped and so we have a set of periodic augmenting paths with fewer (if any) intersections than before. Now it is easy to prove the subclaim.

That the claim is true is also easily established by recalling the details of the Hungarian algorithm [20].

The optimal matching produces a resource assignment whose resource cycle is, in general, a positive multiple of m. Obviously it does not require more resource units over this larger resource cycle than the resource assignment we started with.

The fact that the optimal matching is periodic suffices to prove the theorem. □

In order to grasp the feeling of the theorem, suppose that a specified resource unit is used by a set $\Gamma' \subset \Gamma$ of activities, that is the resource unit is released by some activity in Γ' and then is resumed by some other activity of Γ' and so on, always in a cyclic way, in view of the previous assumption. The time necessary for the resource unit to be used by all activities in Γ', and to be again available for the first one, need not be equal to one period; in fact it can be any value mT with m positive integer. The situation can be viewed as if the resource unit were travelling through the activities of Γ' completing the tour in m periods. If $m > 1$, since each periodic activity must be restarted after one period, there must be another resource unit ready to be used by the first activity in Γ' at the beginning of the second period. In principle there is no reason why this second resource unit travels through the activities in the same sequence as the first unit did one period earlier. What the theorem says is that it is optimal, in terms of number of used resource units, that this second unit (and also all possible other ones) is used by the activities in Γ' in the same sequence as was the first unit.

Therefore m resource units must be circulating in the tour equally spaced of one period. This fact establishes that the number of periods necessary for a set Γ' of activities to be completed by one resource unit is equal to the number of resource units needed by Γ'.

Of course there may be several disjoint subsets Γ_j of activities, each one of them using m_j resource units in a cyclic way. Hence the permutation transforming $RA(p)$ into $RA(p+1)$ is the direct product of some cyclic permutations of cardinality m_j. The resource cycle is then the least common multiple of the m_j's times T. The presence of resources therefore introduces several periods which have to be taken into account simultaneously, and this is the very point where the EPESP model is called for.

Due to the periodic behaviour of the resource units, a resource assignment can be also expressed in terms of resources rather than in terms of activities. In fact, according to the previous results, for each resource R there exists a permutation $PA(R)$ on $\Gamma(R)$ specifying the sequences of activities using the same resource units. The permutation $PA(R)$, being the direct product of some cyclic permutations $PA_1(R), \cdots, PA_{q(R)}(R)$, defines a partition $Q(R) = \{\Gamma_1(R), \cdots, \Gamma_{q(R)}(R)\}$ of $\Gamma(R)$ and to each $Q(R)$ an array $M(R) = \{m_1(R), \cdots, m_{q(R)}(R)\}$ of positive integers is associated such that $\sum_i m_i(R) = |R|$ (again see Fig. 5).

Hence it is possible to express an assignment of resources to activities by specifying $Q(R)$, $M(R)$, and $PA_j(R)$ for each resource R. Indeed this is the way we shall relate resources and activities and this will allow expression of the relationship among resources and periodic events by span constraints of the type described in § 2. With this respect we shall use the following terminology:

A *complete resource plan* is a specification concerning $Q(R)$, $M(R)$, and $PA_j(R)$.

A *partial resource plan* is a specification concerning only $Q(R)$ and $M(R)$.

Of course any resource plan is completely determined by a resource assignment together with an activity schedule, and conversely, any resource assignment is completely determined by a resource plan together with an activity schedule. However, if the activity schedule is not specified neither a resource assignment determines a resource plan nor vice versa.

3.2. Resource planning with a fixed activity schedule. The results of this section can be considered a corollary of Theorem 3. In fact the periodic behaviour of the resource units, implicitly assumed throughout this section, was established by that theorem.

In this section we consider the problem of finding a feasible RA minimizing the number of resource units needed for a fixed activity schedule. As was seen in the previous section, the number of needed resource units is equal to $\sum_j m_j(R)$ for each resource R.

This problem can be solved by finding permutations $PA(R)$ of minimum cost, with the cost directly related to the $m_j(R)$'s. Of course the problem directly splits into separated problems, one for every resource.

For each resource R we build a graph (V, E) where $V = V^- \cup V^+$ with $V^- = \{\varepsilon^-(\gamma) : \gamma \in \Gamma(R)\}$, $V^+ = \{\varepsilon^+(\gamma) : \gamma \in \Gamma(R)\}$, and $E = A \cup B$ with A the set of arcs from V^- to V^+ corresponding to the activities in $\Gamma(R)$, and with B the complete set of arcs from V^+ to V^- corresponding to assignments of one resource unit from one activity to another one. That is, each arc $(\varepsilon^+(\gamma_i), \varepsilon^-(\gamma_j)) \in B$ expresses the fact that resource units may be assigned in sequence to the activities γ_i and γ_j.

Then let us define

$$x(\gamma_i, \gamma_j) = \min_{x \in Z} \{t(\varepsilon^-(\gamma_j)) - t(\varepsilon^+(\gamma_i)) + zT \geqq s(\gamma_i, \gamma_j)\}$$

and let $z(\gamma_i, \gamma_j)$ be the argument yielding the minimum.

To arcs in A the costs $x(\gamma)$ are assigned whereas the costs $x(\gamma_i, \gamma_j)$ are assigned to the arcs in B. The problem is then to find a set $C = \cup C_j$ of arc disjoint directed circuits in (V, E) of minimum cost and such that $A \subset C$. In fact

$$\left(\sum_j m_j\right)T = \sum_A x(\gamma) + \sum_{B \cap C} x(\gamma_i, \gamma_j) = \left(\sum_A z(\gamma) + \sum_{B \cap C} z(\gamma_i, \gamma_j)\right)T.$$

It is immediate to see that this is equivalent to a weighted bipartite matching problem for the complete bipartite graph (V^+, V^-, B) with costs given by the integers $z(\gamma_i, \gamma_j)$. From the optimal matching $\hat{B}$, the permutation $PA(R)$ is immediately derived and obviously

$$\sum_j m_j = \sum_A z(\gamma) + \sum_{\hat{B}} z(\gamma_i, \gamma_j),$$

with

$$m_j = \sum_{A \cap C_j} z(\gamma) + \sum_{\hat{B} \cap C_j} z(\gamma_i, \gamma_j).$$

So the solution of the problem consists of several resource cycles C_j (possibly just one circuit) each associated to a certain multiple m_j of the basic period T. It may be interesting to note that if we want a solution consisting of one resource cycle, the problem is no longer a weighted assignment but it becomes a TSP. On the other hand if we want a solution in which each resource cycle is long T, a circular graph coloring problem is produced which is also NP-complete [11]. Note also that if $x(\gamma_i) < T$, for all i and $s(\gamma_i, \gamma_j) = 0$, for all i, j, then $z(\gamma_i, \gamma_j) \in \{0, 1\}$ and so the weighted bipartite matching problem actually becomes a simpler cardinality problem.

3.3. Activity scheduling with a fixed complete resource plan. The problem considered in this section is the one of scheduling a given set of periodic events subject to some span constraints of the type described in § 2.1, with the additional condition that some (or even all) events are starting and/or ending events of activities for which a complete resource plan has been fixed and is required to be feasible. To say that a resource plan has been fixed means that for each resource R the partition $Q(R)$, the permutation $PA(R)$ and the arrays $M(R)$ are fixed to some definite values.

In order to have a feasible resource assignment, we have to add the condition that the events referring to the activities in $\Gamma_j(R)$ have to be scheduled in sequence with respect to $m_j(R)T$ (see §§ 2.5 and 2.6), according to the sequence given by $PA_j(R)$. So it is just a matter of applying the results of § 2.6. To be more specific, let $\Gamma_j(R) =$

$\{\gamma_1, \cdots, \gamma_p\}$ with activities labelled according to the sequence. For the sake of simplicity let us assume that the activity durations $x(\gamma_i)$ are fixed, so one periodic event (say the starting event) ε_i is sufficient to identify the activity γ_i. Then the periodic events $\varepsilon_1, \cdots, \varepsilon_p$ have to be scheduled in sequence with respect to $m_j(R)T$.

Furthermore, feasibility of the resource assignment is obtained by imposing, for each pair of successive events,

$$a_i := (\varepsilon_i, \varepsilon_{(i+1) \bmod p}) \qquad i = 1, \cdots, p,$$

and also imposing the span constraint given by

$$v(a_i) \in [x(\gamma_i) + s(\gamma_i, \gamma_{(i+1) \bmod p}) + zm_j(R)T,\ W + x(\gamma_i) + s(\gamma_i, \gamma_{(i+1) \bmod p}) + zm_j(R)T]$$

for some $z \in Z$, where the quantities $s(\gamma_i, \gamma_j)$ have been defined in § 3.1 and W is computed as

$$W := m_j(R)T - \sum_{i=1}^{p} x(\gamma_i) + s(\gamma_i, \gamma_{(i+1) \bmod p}).$$

The above span constraints clearly subsume the ones relative to the sequencing condition.

This is an EPESP with the added feature of sequencing (an example is shown in Fig. 6). Therefore a possible algorithm to solve it is the one described in § 2.5 with the modification pointed out in § 2.6. There is no conceptual difficulty in dealing with variable activity durations; it is just a matter of considering the ending events as well and of sequencing the events in the natural way with possible span constraints between the starting and the ending events.

Some particular cases are worth mentioning. If $|\mathscr{R}(\gamma)| \leqq 1$, that is, if each periodic activity requires at most one resource, then the sequence circuits are disjoint and therefore, since the span constraints involving multiples of the period T refer to arcs in the sequence circuits, the tree integrality condition expressed in §§ 2.3 and 2.5 is automatically satisfied for any spanning tree containing all arcs but one of every sequence circuit.

If $|\Gamma(R)| = 2$, that is only two periodic events, corresponding to the starting events of two activities γ_i, γ_j, have to be sequenced, the following span for $(\varepsilon^-(\gamma_i), \varepsilon^-(\gamma_j))$ suffices to model a feasible resource assignment

$$[x(\gamma_i) + s(\gamma_i, \gamma_j),\ mT - x(\gamma_j) - s(\gamma_j, \gamma_i)]_{mT}.$$

In Fig. 6 an example is given with seven periodic activities of fixed duration and no set-up times. Three resources are involved consisting of three, two, and three units respectively (labeled r_1–r_8). The first four activities need the first and third resource (i.e., any one unit of both the first and the third resource), and the remaining activities need the second and the third resource. These are the problem data for which a feasible activity schedule together with a feasible resource assignment is looked for.

We suppose that the resource plan has been fixed in the following way: the resource cycle of resource R_1 consists of one cycle, namely $\gamma_1 \to \gamma_2 \to \gamma_3 \to \gamma_4 \to \gamma_1$, lasting three periods since $|R_1| = 3$. Correspondingly, four arcs are given on the network with span $[x(\gamma_i), W + x(\gamma_i)]_{30}$ for the arc outgoing from node i with $W = 15$, i.e., the period minus the sum of the durations along the cycle. Similarly the resource cycle of resource R_2 consists of the single cycle $\gamma_5 \to \gamma_6 \to \gamma_7 \to \gamma_5$ lasting two periods and, correspondingly, three arcs are given on the network with appropriate spans.

The resource R_3 is used in a different way: its three units go over three independent cycles, necessarily lasting one period. In particular one of the cycles (there is no need of explicitly assigning cycles to resource units since these are indistinguishable) involves

$\Gamma = \{\gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5, \gamma_6, \gamma_7\}$
$x(\gamma_1)=4, \quad x(\gamma_2)=3, \quad x(\gamma_3)=5, \quad x(\gamma_4)=3, \quad x(\gamma_5)=4, \quad x(\gamma_6)=5, \quad x(\gamma_7)=4$
$s(\gamma_i, \gamma_j)=0$
$\mathscr{R} = \{R_1, R_2, R_3\}$
$R_1=\{r_1, r_2, r_3\}, \quad R_2=\{r_4, r_5\}, \quad R_3=\{r_6, r_7, r_8\}$
$\mathscr{R}(\gamma_1)=\mathscr{R}(\gamma_2)=\mathscr{R}(\gamma_3)=\mathscr{R}(\gamma_4)=\{R_1, R_3\}$
$\mathscr{R}(\gamma_5)=\mathscr{R}(\gamma_6)=\mathscr{R}(\gamma_7)=\{R_2, R_3\}$
$\Gamma(R_1)=\{\gamma_1, \gamma_2, \gamma_3, \gamma_4\}, \quad \Gamma(R_2)=\{\gamma_5, \gamma_6, \gamma_7\}, \quad \Gamma(R_3)=\Gamma$

FIG. 6.1. *Activity data.*

$Q(R_1)=\{\Gamma(R_1)\}, \quad Q(R_2)=\{\Gamma(R_2)\}, \quad Q(R_3)=\{\{\gamma_1, \gamma_6\}, \{\gamma_2, \gamma_7, \gamma_4\}, \{\gamma_5, \gamma_3\}\}$
$PA(R_1)=(1, 2, 3, 4), \quad PA(R_2)=(5, 6, 7), \quad PA(R_3)=(1, 6)(2, 4, 7)(5, 3)$ (cyclic notation)
$m(R_1)=3, \quad m(R_2)=2, \quad m_1(R_3)=m_2(R_3)=m_3(R_3)=1$

FIG. 6.2. *Complete resource plan.*

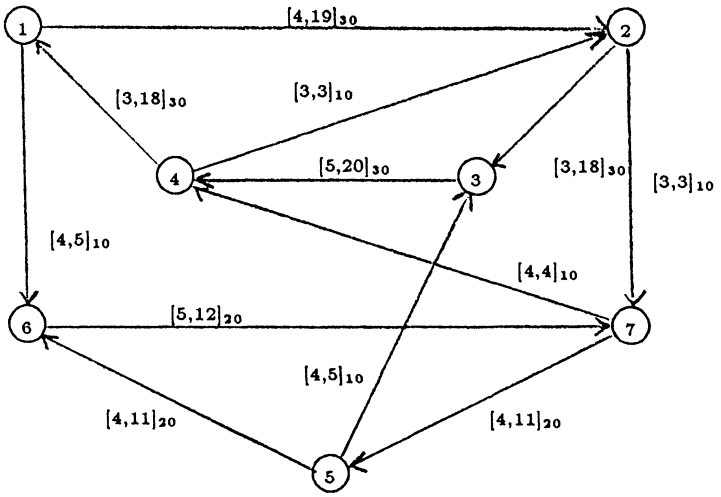

FIG. 6.3. EPESP *network for activity scheduling with fixed resource plan.*

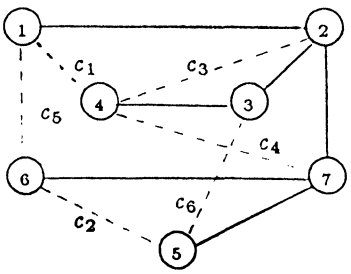

FIG. 6.4. *Spanning tree satisfying tree integrality.*

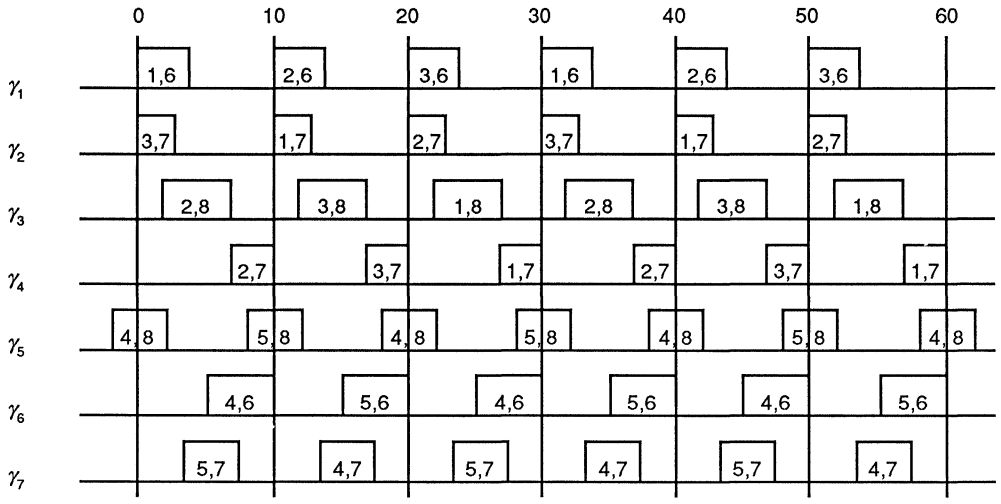

FIG. 6.5. *Solution.*

the activities $\gamma_2 \rightarrow \gamma_7 \rightarrow \gamma_4 \rightarrow \gamma_2$ (in this order) raising the corresponding arcs and spans on the network. The other two cycles involve only two activities each, namely (γ_1, γ_6) and (γ_3, γ_5). Since two activities are always sequenced only one arc between the activities suffices to model a feasible resource assignment, as was already pointed out.

The resulting network is shown in Fig. 6.3. A spanning tree satisfying tree integrality is shown in Fig. 6.4. In problems involving sequence conditions it is not convenient to sort the chords for increasing span lengths; in fact, since some span constraints have a fixed value for z due to the sequence condition $\sum z_i = -1$, (and hence they correspond to a level with one successor only in the search tree) it is more convenient to have these span constraints at the highest possible levels. So in the example c_1 and c_2 are the chords closing the cycles of resources R_1 and R_2, respectively. In order to fulfill the sequence condition, since all arcs of the cycle but one belong to the spanning tree in both cases,

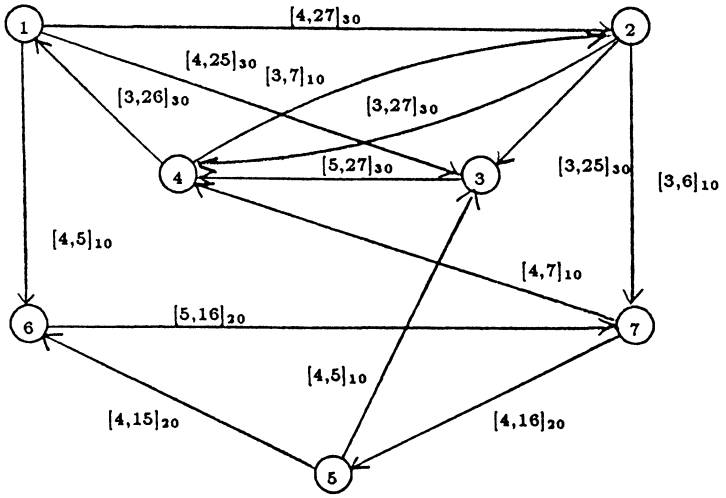

FIG. 6.6. EPESP *network for activity scheduling with fixed partial resource plan.*

it suffices to set $D(c_1) = [-27, -12]$ and $D(c_2) = [-16, -9]$ which correspond to the condition $\sum z_i = -1$.

As far as the resource cycle $\gamma_2 \rightarrow \gamma_7 \rightarrow \gamma_4 \rightarrow \gamma_2$ is concerned, only one arc of the cycle belongs to the spanning tree with value $z = 0$. So it is required $z(c_3) + z(c_4) = -1$. It can be seen that when the search tree level corresponding to c_3 is reached a first guess $z(c_3) = -2$ is made. This implies that in the next level $z(c_4) = 1$. It turns out that this value yields a feasible potential and so the visit of the search tree may proceed to deeper levels. For this particular example the final solution is obtained without backtracking. It is displayed in the form of a GANTT diagram in Fig. 6.5; the numbers inside the boxes refer to the resource units.

3.4. Activity scheduling with a fixed partial resource plan. A drawback of the model described in the previous section is that a very detailed description of the resource plan is required to be given in advance in order to schedule all events. In practical applications any resource plan is all right as long as it is feasible and consistent with the available number of resource units. Therefore we describe in this section a model which is relaxed with respect to the input data requirement but is computationally more expensive.

With respect to the problem of the previous section, the way the activities in each $\Gamma_j(R)$ have to be sequenced is no longer part of the problem data but a variable to be determined. Let us suppose again for the sake of simplicity that the activity durations are fixed so that the starting events $\varepsilon_1, \cdots, \varepsilon_p$ identify the activities in $\Gamma_j(R)$. Then the following span constraints between *all* (unordered) pairs $\{\varepsilon_h, \varepsilon_k\}$, $h = 1, \cdots, p-1$, $k = h+1, \cdots, p$

$$[x(\gamma_h) + s(\gamma_h, \gamma_k), mT - x(\gamma_k) - s(\gamma_k, \gamma_h)]_{mT}$$

correspond to the requirement of a feasible resource assignment.

In terms of the network model of the EPESP, these span constraints form a certain number of cliques. Scheduling the activities on each clique is a problem comparable to a TSP as seen in § 2.7. Furthermore, all these subproblems interact via other span constraints. It is clear that the problem of scheduling the activities with a partial resource plan is a very hard one unless the cliques are rather small. Compare, for instance, the network in Fig. 6.6 with the one in Fig. 6.3; both refer to the same problem data, but in Fig. 6.6 the network has been drawn by fixing the resource plan only partially. This relaxed requirement has the effect of increasing the number of arcs (only by two in this case due to the small cliques) and the span lengths.

Therefore, in order to devise sensible strategies to solve the problem of scheduling the activities with a fixed partial resource plan, it may be convenient to exploit the problem of § 3.3 as a subproblem to be used for a branch and bound strategy. We recall that in the problem of § 3.3 a complete resource plan is assumed, that is, the sequencing of the activities on each clique is kept fixed. TSP techniques may be proper tools to assess convenient sequences. In this sense especially the problem of scheduling activities with a complete resource plan becomes a sensible and useful model.

3.5. Activity scheduling with a fixed number of resource units. In certain practical applications only the global number of available resource units is prescribed in advance and in general any partition $Q(R)$ is accepted as long as it makes the schedule feasible.

Unfortunately the requirement on the use of the resources is so relaxed in this problem that it is not possible to model it within the framework of span constraints as we did for the problems in the previous sections. It is certainly out of question to check all possible partitions $Q(R)$ until feasibility is found. Nevertheless the following considerations can provide some help.

First of all note that if $|R| = 1$ for all resources then only the trivial partition $Q(R) = \{\Gamma(R)\}$ has to be considered, with the obvious value $m(R) = 1$. In this case the problem is exactly the one of scheduling the activities with a fixed partial resource plan, which can be solved through PESP.

If on the contrary there are several resource units of each resource, then a favourable situation is faced if $s(\gamma_i, \gamma_j) = 0$ for all i and j. In fact let us suppose that an activity schedule is given and that the goal is the minimization of the needed resource units. Let us consider the following two alternatives: either we apply the results of § 3.2 and solve an assignment problem or we add the extra requirement of having an assignment corresponding to a cyclic permutation, which amounts to solving a TSP. According to [16 p. 99], the difference between the two optimal values for this particular set of costs (in fact a graded matrix of 0's and 1's) is at most 1.

Therefore if we want to find a feasible activity schedule with a fixed number m of available resource units of some resource R, we may consider the following approach:

1) find an activity schedule with the following fixed partial resource plan

$$Q(R) = \{\Gamma(R)\} \qquad m(R) = m.$$

If there exists a feasible solution, stop, otherwise

2) find an activity schedule with the following fixed partial resource plan

$$Q(R) = \{\Gamma(R)\} \qquad m(R) = m + 1.$$

If there is no feasible solution, stop, because the original problem does not have feasible solutions either, otherwise,

3) minimize the number $\hat{m}$ of needed resource units for the activity schedule just found. If $\hat{m} \leqq m$, stop, otherwise the situation is rather hopeless.

Special techniques could be conceived to handle the last point, but this is beyond the scope of this paper.

4. Applications.

4.1. Periodic Job-Shop. Let us first recall the usual (aperiodic) Job-Shop Problem (which will be referred to in the following as JSP). There is a set $\mathcal{J}$ of *jobs*. For each job $J \in \mathcal{J}$ there is specified a set of *activities* (or *operations*) $\gamma(J, k), k = 1, \cdots, q(J)$, to be executed on a certain *piece* $r(J)$, in the sequence given by the index k. This way, a one-to-one association between jobs and pieces is given. Each activity γ has a certain given duration $x(\gamma)$.

There is a set $\mathcal{M}$ of *machines*. Each activity γ is assigned to a definite machine $M(\gamma) \in \mathcal{M}$.

The problem is the one of scheduling all activities such that

1) The activities of the same job are scheduled according to the prescribed sequence;
2) The same machine executes at most one activity at a time;
3) A prescribed criterion for the completion time has to be satisfied.

In the Periodic Job-Shop Problem (PJSP) each activity has to be repeated at regular intervals of length T, so it has to be considered a periodic activity. Since its duration is fixed, one periodic event identifies each periodic activity. In the periodic case each job J describes a set of activities to be executed on infinitely many identical pieces $\cdots, r_1(J), r_2(J), \cdots$.

The pieces $r_i(J)$ enter the job-shop at regular intervals of length T and remain in it for a certain time $T(J)$, which is the same for all of them. Let $(m(J) - 1)T < T(J) \leqq m(J)T$ for a certain integer $m(J)$. Then we identify the pieces $r_i(J)$ and $r_{i+m(J)}(J)$ in

order to model this problem in our framework. This corresponds to having a finite set $R(J) := \{r_1(J), \cdots, r_{m(J)}(J)\}$ of pieces. In the framework of § 3.1, $R(J)$ is a particular resource, and we consider the pieces as resource units. Through the above identification an open system is transformed into a closed (cyclic) one.

Each machine is a resource as well. In particular it is a resource consisting of a single unit.

Each set $\mathscr{R}(\gamma)$ consists of two elements, i.e.,

$$\mathscr{R}(\gamma(J, k)) = \{R(J), M(\gamma(J, k))\}$$

and obviously

$$\Gamma(R(J)) = \{\gamma(J, 1), \cdots, \gamma(J, q(J))\}$$

$$\Gamma(M) = \{\gamma(J, i): M = M(\gamma(J, i))\}.$$

Then the problem is scheduling the periodic activities such that:

1) The resource plan concerning the resources $R(J)$ (pieces) is *completely* specified as follows: Due to the concept of "job," the partitions $Q(R(J))$ are the trivial partitions $Q(R(J)) = \{\Gamma(R(J))\}$ and the permutations $PA(R(J))$ must be cyclic, corresponding to the sequence $\gamma(J, 1), \cdots, \gamma(J, q(J))$, which are obviously assigned a priori by technological requirements. Concerning the integers $m(J)$ assigned to each job, they are not actually imposed a priori and constitute a set of design variables subject to the only obvious limitation such that

$$m(J)T \geqq \sum_k x(\gamma(J, k)) + s(\gamma(J, k), \gamma(J, k+1)).$$

At the moment they are arbitrarily specified by the designer. We shall comment on this in a few paragraphs.

2) The resource plan concerning the resources M (machines) is *partially* specified as follows: Due to the fact that each machine is considered different from all other ones and hence each activity is assigned to one definite machine, the partitions $Q(M)$ have to be the trivial partitions $\{\Gamma(M)\}$ and $m(M)$ must be equal to one. The permutations $PA(M)$ must be cyclic, but they are not specified and have to be determined in order to find a feasible schedule.

3) In the PJSP there is no completion time of course. A related measure of productivity is given by the number of processed pieces leaving the job-shop in a time unit, i.e., by $|\mathscr{J}|/T$. So, if $|\mathscr{J}|$ is considered fixed and T is not imposed a priori by some other constraints, we might ask for the minimal T realizing a feasible schedule, a task which could be performed via a bisection procedure.

Therefore a PJSP can be formulated within the framework of the model developed in § 3 and solved via the techniques described in § 2.

Some other features of the PJSP are worth mentioning. Let $\hat{T}$ be the minimal T realizing a feasible schedule. First just note that an obvious lower bound for $\hat{T}$ is given by

$$\max\left\{\max_{M \in \mathscr{M}} \sum_{\gamma \in \Gamma(M)} x(\gamma); \max_{J \in \mathscr{J}} \sum_{\gamma \in \Gamma(J)} x(\gamma)\right\}.$$

The schedule obtained with $\hat{T}$ exhibits a critical circuit in the network model of PESP in the sense that all span constraints of the circuit are satisfied exactly at the span boundaries. This situation is the exact counterpart of the critical path for the JSP.

The concept of "selection" of a machine for the JSP, that is, prescribing the precedence order of the activities on the same machine, is translated into PJSP as a sequence

specification and therefore, if all machines are given a particular selection, all resource plans are completely specified and the problem of scheduling the activities is solved as described in § 3.3.

An interesting feature of the PJSP, not present in the JSP, consists in the fact that all pieces of a certain job J remain in the job-shop for an amount of time given by $T(J)$. It is important to note that increasing $T(J)$ (while keeping T fixed) has the effect of relaxing the sequence constraints of the problem to such an extent that for sufficiently large $T(J)$ any resource assignment can be made feasible. However, large $T(J)$ result in a larger number of pieces simultaneously present in the job-shop and therefore produce storage problems and raise the opportunity costs associated with the capital committed to the material under process.

Thus two different objectives have been identified: 1) A productivity rate associated with the period T and the number of jobs $|\mathcal{J}|$; 2) Opportunity costs associated with the $T(J)$'s and therefore with the $m(J)$'s.

As a further reference on this problem see [26]. For an example reconsider Fig. 6 where two jobs can be specified by the activities $\gamma_1, \cdots, \gamma_4$ and $\gamma_5, \cdots, \gamma_7$ respectively and the three resource units r_6, r_7, r_8 can be reconsidered as three different machines, i.e., three different resources, each one of them consisting of a single resource unit.

An extension of the PJSP could take into account the pallet scheduling in case the motion of pieces is performed automatically through pallets. If the pallets are a scarce resource they could be considered an additional resource. Then it is just a matter of considering new activities corresponding to pallet motions between two original activities. The resulting problem could be very difficult. A more realistic approach would be to disregard the pallet problem in the first stage of scheduling and then, once the activity schedules are fixed, minimize the number of needed pallets through the bipartite matching problem of § 3.2.

4.2. Transportation scheduling. In transportation one typical problem is of minimizing the number of vehicles needed to implement a given time schedule. This was solved previously by Dilworth in the aperiodic case. The periodic case has also been solved via different approaches both when deadheading is allowed and when it is not [12], [13], [18]. We recall that deadheading is the transfer of a vehicle from one terminal to another one with a trip out of schedule.

The model presented in this paper provides a rather simple framework to solve the problem of minimizing the number of vehicles with no conceptual difference between allowing deadheading or not. In this model each trip is considered as an activity γ. Then we may define the setup times $s(\gamma_1, \gamma_2)$ to be the minimal time needed for a vehicle to be available for trip γ_2 after completion of trip γ_1. If deadheading is allowed $s(\gamma_1, \gamma_2)$ takes care of the actual time needed for transferring the vehicle between two distant terminals, otherwise $s(\gamma_1, \gamma_2)$ can be simply set to infinity.

As shown in § 3.2 this is a weighted assignment and this conclusion agrees with previous results [12], [13], [15], [18].

An interesting extension consists in allowing some flexibility in the time schedule within some prescribed tolerances. So it is possible to reschedule the trips according to the results of § 3.3 given a certain complete resource plan. It is beyond the scope of this paper to give methods to derive a complete resource plan in order to find the periodic trip schedule. Actually, by combining our model with other models it is possible to develop efficient heuristics. This is a matter of current research.

4.3. Traffic light scheduling. This application deals with the problem of scheduling the switching times of green and red lights in a system of signals controlling urban traffic.

We look for a periodic schedule with a given period T. This is a common practice in several applications at least for time intervals much larger than the period.

We may model the problem by considering a green time as a periodic activity γ. Therefore there are as many activities as the independent signals. For the sake of simplicity we suppose that the green light durations $x(\gamma)$ are given a priori; so one periodic event $\varepsilon(\gamma)$, say the starting of green light, identifies each periodic activity. A more complex model involving variable green light durations can be found in [25].

The periodic activities, i.e., the green times, are subject to the following constraints: two signals (γ_1, γ_2) controlling the access to the same physical area cannot be green simultaneously. So, by also taking into account proper clearance times $s(\gamma_1, \gamma_2)$ and $s(\gamma_2, \gamma_1)$, the following span constraint must be obeyed by the two signals γ_1 and γ_2:

$$\tau(\varepsilon(\gamma_2)) - \tau(\varepsilon(\gamma_1)) \in [-x(\gamma_1) - s(\gamma_1, \gamma_2),\ x(\gamma_2) + s(\gamma_2, \gamma_1)]_T.$$

Besides this type of constraint we may also model a coordination between signals controlling the same traffic flow, by imposing that the switching of a downstream signal must occur within a given time interval $[\delta^-, \delta^+]$ from the switching of the corresponding upstream signal. So PESP is enough to model this problem.

5. Conclusions. A general framework to deal with a large class of scheduling problems in a periodic environment has been proposed.

First the case involving only separate periodic events with constraints involving the relative position of pairs of them has been considered. It has been shown to be NP-complete and some other properties have been settled. An algorithm for it has been devised, which exhibits a satisfactory average performance in practical cases. Some extensions have also been encompassed, such as multiperiod situations.

Then the concepts and methods for periodic event scheduling problems have been exploited to deal with periodic activities using resources. Some different classes of problems have been classified and the relationships among them have been analyzed in a unitary approach.

Finally, some specific applications have been considered in detail. In the first one a periodic version of the well-known Job-Shop Scheduling problem has been treated. The second one dealt with vehicles scheduled to meet a given time table, possibly with some assigned tolerances. The last one considered setting traffic lights controlling urban traffic while complying with simple safety requirements. Each of the above problems has been modelled as a specific periodic activity scheduling problem with resources.

In conclusion, it may be pointed out that this should not be intended as a conclusive work on periodic scheduling issues. Rather, it should stimulate further research effort on such topics. For instance it is worth mentioning the possibility of refining the algorithmic aspects, for instance adapting them to specific problem features in order to achieve an improved computational efficiency. Another challenging subject contemplates optimization problems instead of only feasibility ones.

REFERENCES

[1] K. R. BAKER, *Scheduling a full-time workforce to meet cyclic staffing requirements*, Management Sci., 20 (1974), pp. 1561–1568.

[2] J. J. BARTHOLDI III, *A guaranteed-accuracy round-off algorithm for cyclic scheduling and set covering*, Oper. Res., 29 (1981), pp. 501–510.

[3] J. J. BARTHOLDI III, J. B. ORLIN, AND H. D. RATLIFF, *Cyclic scheduling via integer programs with circular ones*, Oper. Res., 28 (1980), pp. 1074–1085.

[4] T. E. BARTLETT, *An algorithm for the minimum number of transport units to maintain a fixed schedule*, Naval Res. Logist. Quart., 4 (1957), pp. 139–149.

[5] S. E. BECHTOLD, *Work-force scheduling for arbitrary cyclic demands*, J. Operations Management, 1 (1981), pp. 205–214.
[6] R. E. BURKARD, *Optimal schedules for periodically recurring events*, Discrete Appl. Math., 15 (1986), pp. 167–180.
[7] A. CEDER AND H. I. STERN, *Deficit function bus scheduling with dead-heading trip insertions for fleet size reductions*, Transpn. Sci., 15 (1981), pp. 338–363.
[8] W. DAUSCHA, H. D. MODROW, AND A. NEUMANN, *On cyclic sequence types for constructing cyclic schedules*, Zeit. f. Oper. Res., 29 (1985), pp. 1–30.
[9] H. EMMONS, *Work-force scheduling with cyclic requirements and constraints on days off, weekends off, and work stretch*, IIE Trans., 17 (1985), pp. 8–16.
[10] S. FRENCH, *Sequencing and Scheduling*, Ellis Horwood, Chichester, United Kingdom, 1982.
[11] M. R. GAREY, D. S. JOHNSON, G. L. MILLER, AND C. H. PAPADIMITRIOU, *The complexity of coloring circular arcs and chords*, SIAM J. Algebraic Discrete Methods, 1 (1980), pp. 216–227.
[12] I. GERTSBAKH AND Y. GUREVICH, *Constructing an optimal fleet for a transportation schedule*, Transpn. Sci., 11 (1977), pp. 20–36.
[13] ———, *Homogeneous optimal fleet*, Transportation Res., 16B (1982), pp. 459–470.
[14] W-L. HSU, *On the general feasibility test of scheduling lot sizes for several products on one machine*, Management Sci., 29 (1983), pp. 93–105.
[15] V. B. KATS, *An exact optimal cyclic scheduling algorithm for multioperator service of a production line*, Automat. Remote Control, 43 (1982), pp. 538–542.
[16] E. L. LAWLER, J. K. LENSTRA, A. H. G. RINNOOY KAN, AND D. B. SHMOYS, eds., *The Traveling Salesman Problem*, John Wiley, Chichester, United Kingdom, 1985.
[17] M. O'hEIGEARTAIGH, J. K. LENSTRA, AND A. H. G. RINNOOY KAN, eds., *Combinatorial Optimization: Annotated Bibliographies*, John Wiley, Chichester, United Kingdom, 1985.
[18] J. B. ORLIN, *Minimizing the number of vehicles to meet a fixed periodic schedule: an application of periodic posets*, Oper. Res., 30 (1982), pp. 760–776.
[19] J. B. ORLIN, M. A. BONUCCELLI, AND D. P. BOVET, *An $O(n^2)$ algorithm for coloring proper circular arc graphs*, SIAM J. Algebraic Discrete Methods, 2 (1981), pp. 88–93.
[20] C. H. PAPADIMITRIOU AND K. STEIGLITZ, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, New Jersey, 1982.
[21] K. S. PARK AND D. K. YUN, *Optimal scheduling of periodic activities*, Oper. Res., 33 (1985), pp. 690–695.
[22] A. H. G. RINNOOY KAN, *Machine Scheduling Problems*, Martinus Nijhoff, The Hague, 1976.
[23] R. T. ROCKAFELLAR, *Network flows and monotropic optimization*, John Wiley, New York, N.Y., 1984.
[24] P. SERAFINI AND W. UKOVICH, *An approach towards solving periodic scheduling problems*, Ricerca Operativa, 35 (1985), pp. 17–39.
[25] ———, *A mathematical model for the fixed-time traffic control problem*, European J. Oper. Res., 41 (1989), pp. 1–14.
[26] ———, *Decomposing production scheduling problems in a periodic framework*, in Proceedings IXTH International Conference on Production Research, Cincinnati, OH, August 1987.
[27] H. S. STONE AND P. SIPALA, *The average complexity of depth-first search with backtracking and cutoff*, IBM J. Res. Develop., 30 (1986), pp. 242–258.
[28] A. TUCKER, *Coloring a family of circular arcs*, SIAM J. Appl. Math., 29 (1975), pp. 493–552.
[29] R. R. VEMUGANTI, *On the feasibility of scheduling lot sizes for two products on one machine*, Management Sci., 24 (1978), pp. 1668–1673.