Transactions on Information and Communications Technologies vol 1, © 1993 WIT Press, www.witpress.com, ISSN 1743-3517

S.O.S. a shell for opportunistic scheduling in manufacturing environments

E. Paolucci,[a] G. Gini,[b] M. Possa & S. Preda
[a] *Department of Economics and Production, Politecnico di Milano, P.zza L. Da Vinci 32 20133 Milano, Italy*
[b] *Department of Electronics and Information, Politecnico di Milano*

ABSTRACT

The work describes a scheduling system based on an opportunistic framework. It exploits the potentiality of a predictive module that provides a-priori information about the conflicts that may arise in a schedule, managing a constraint based approach in problem solving. The system SOS has been designed and realized to apply in real productive environments this theoretical approach. Requirements of efficiency and flexibility of use have been particularly considered.

GUIDELINES AND SCHEDULING STRATEGIES IN SOS

In this paper we will present an approach to scheduling problems which is focused on the requirements of real manufacturing environments. Therefore we considered aspects that are tied to the managerial perspective of the problem. The SOS system is designed both for computing solutions in scheduling problems and for supporting manager in obtaining schedules that reflect the state of shop-floor and organizational problems in production.

Two aspects have been considered in these perspective:

- it develops a temporal predictive method [7] to support a preventive analysis of activity interactions and to detect bottlenecks. It exploits an a-priori analysis of the effects of organizational constraints on the scheduling problem, evaluating alternatives as quick as possible.

- If some unexpected event happens on the job shop, it works as a support tool for the evaluation of different reaction decisions (reactive scheduling [4]).

The contribution of AI-based techniques is focused especially on the analysis of the effects that scheduling variables have on solutions and on the definition of adequate reasoning patterns.

We based our approach on opportunistic reasoning [5]. This term characterizes a problem solving process in which the focus of attention is constantly directed to those choices that appear the most promising for reaching a solution. This policy is expected to reduce the search space (and therefore the computational complexity), focusing the reasoning on areas that are considered as the more constrained (island driving).

The capability to provide a feasible solution in a sufficiently short time is fundamental for manufacturing environments, but we must remember that the existence of at least one feasible solution is not always guaranteed. The evaluation of effects of constraints becomes fundamental to identify a solution. Production managers want to know how the decision to change or to relaxing the value of some constraints (concerning for example orders due-dates or priorities) affects the quality of final schedules and the global productive efficiency. This feature is particularly important when a feasible solution does not exist; in this case, from a managerial viewpoint, it is required to the system to support the user in "building" a feasible solution.

From a mathematical perspective we considered scheduling as a Constraint Satisfaction Problem (CSP) [6], concerned with the assignment of values to variables subject to a set of constraints. The solution of the CSP (a feasible set of values assumed by the variables) depends on the constraints defined on the problem: by changing the constraints, the solution will change, too. In solving a CSP, two decisions have to be made at each cycle, i.e. which variable to instantiate next and which value to assign to that variable. This formalization highlights the importance associated to constraints analysis:

- they allow the quick computation of a feasible solution, without exploring the entire problem domain.

- Through their analysis we can obtain information about how they can be modified opportunistically during scheduling.

We distinguished between two classes of constraints [3], considering the possibility for production managers to change their values in a very short time horizon:

Restrictions: they must be always respected and cannot be easily and quickly changed; they can be furtherly divided into causal constraints (they represent conditions to be satisfied before starting an activity), physical constraints (each equipment has specific capabilities that restrict the type and the speed of operations it can accomplish), and resources unavailability.

Preferences: restrictions leave scheduling problem underconstrained. Preferences represent the set of managerial constraints and the definition of their values is subject to a decision process. Some examples are: defining orders due date and priority, meeting due-dates, minimizing WIP, maximizing resource utilization, the definition of operational preferences (an order may follow alternative production paths in shop floor), etc.

The role of constraints can be underlined considering a second classification, which concerns the links among activities. In a manufacturing environment orders (made of many activities) are processed contemporaneously; moreover many of them may require the same productive resources. These connections cause two kinds of constraints [9]:

- **intra-order**: they represent explicit links existing among activities belonging to an order (for example temporal relations [1]. These constraints are explicitly defined in the problem.

- **Inter-order**: they represent implicit links existing among activities belonging to different orders. For example activities belonging to different orders may require contemporaneously the same productive resource. These constraints remain implicit after the problem definition.

THE SOS SCHEDULING PROCESS

The goal of our system is to find solutions that satisfies a given set of constraints as quick as possible. If a solution does not exist, the system relaxes only preferences, minimizing the overall effects on final schedule. The scheduling policies followed in SOS can be summarized as follows:

1. Scheduling is made allocating one activities at a time, consider at each step the effects of intra- and inter-order constraints;

2. Most constrained and least-impact policies are implemented at each step. First the most-constrained policy selects dynamically on which agent must be focused scheduling attention. Then, the least impact policy chooses for that agent a value whose impact on the rest of the non-scheduled agents is as small as possible.

   The strategic goal is the identification of critical activities that heavily rely on the possession of highly contended temporal intervals or resources because of intra-order and inter-order interactions (**look-ahead strategy**).

These two policies are based on numeric indexes which account for the particular structure of a problem. They give a measure of the interaction among activities and resources in terms of **variable looseness** (it is a measure of constraint degree) and **value goodness** (it is a measure of the impact of a scheduling decision on non-scheduled activities) [9].

Finally we decided to avoid the utilization of backtracking in SOS systems because of the particular field of application that we identified. In real problems, we have no a priori guarantees concerning the existence of a solution: in this case backtracking is completely useless. Moreover, in order to meet our basic requirements, we prefer to look for a satisfying solution in a short time, instead than to wait for the best solution (which may not exist).

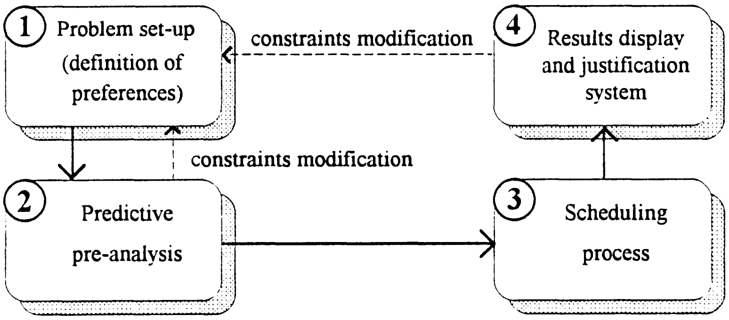

Figure 1. The four steps of SOS system

Figure 1 shows the various scheduling phases in SOS system. A schedule is calculated through four steps which will be described in the next sections.

THE PROBLEM DEFINITION

Using a graphic interface the user of SOS can define the problem characteristics (problem set-up). The user can specify the number of orders to be scheduled, the number of activities that compose each order, the temporal relations among activities, resources required and so forth.

We adopted a temporal representation of relations among activities that holds both symbolic and numeric information: the Temporal/Capacity Constraint Graph (TCCG) [9]. This model consolidate in a graph all the activities composing an order. A node of the graph represents an activity (A_i), and an arc is a temporal link (L_i) between two activities. Loops are not introduced and each link has a direction that originates temporal relations. A set of orders creates a scheduling job.

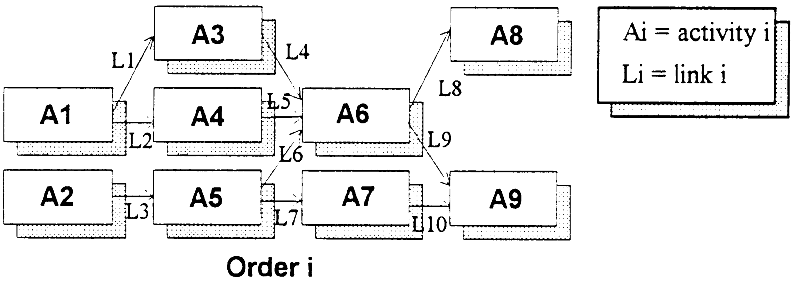

Figure 2. A hypothetical TCCG graph for one order

From a temporal perspective the user can set the following parameters:

- ST_i is the earliest start time for the activity (the start time of the feasible temporal window);
- ET_i is the latest end time for the activity (the end time of the feasible temporal window);
- D_i is the activity duration, with $D_i < ET_i - ST_i$.

The interval (ET_i - ST_i) identifies the temporal window, where an activity must be scheduled. But sometimes the scheduling process can relax this preferential directive.

A temporal link between two activities A_i and A_j is represented by two values [2]:

- INT: it is the minimum time interval between the end of activity A_i, and the start time of activity A_j;
- EXT: it is the maximum time interval between ST_i and ET_j.

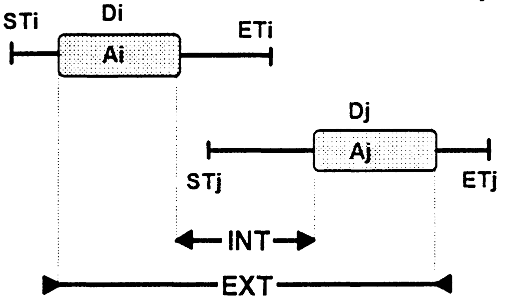

Figure 3. A link between two sequential activities A_i and A_j in order O_k

This kind of representation is at the same time flexible and simple to be used for analyzing relations among temporal constraints. On the other hand, it can model all the temporal relations between two activities [1].

THE PRE-ANALYSIS PHASE

This phase calculates some predictive indexes, derived from PREDIT system [7]. It supplies information for scheduling process that are useful for supporting the computation of the final schedule, or to help the user in a pre-analysis of the influence of constraints.

These indexes point out the most constrained objects (resources and activities) and point out scheduling decisions that have least impact on future steps. Their goal is to speed up scheduling time, minimizing the need of backtracking. These indexes are discreet in time and depend on the division of the temporal axis that we make: the smaller is the temporal sampling quantum, the more exact will be the value of the index (naturally at the cost of reducing the computation speed).

The indexes have the following meaning:

- PST_i (Preferred Start Time of activity A_i): it points out for every activity, the start time (contained in the feasible windows) that minimizes its interactions with other activities belonging to the same order.

- V_i (Slack Index of activity A_i): it measures the interaction degree of an activity with the other activities belonging to its order. It may assume values ranging from zero to one, where the value zero means the absence of interaction with other activities and the value one the absence of slack.

- ID_k (Individual Demand for resources R_k): it measures the demand degree of activity A_i for resource R_k. It is calculated on the basis of PST_i and is given as a function of time.

- AD_k (Aggregate Demand for resource R_k): it is the sum of all the individual demands for resource R_k. It points how contended is resource R_k and instants where this contention is particularly high (bottle-necks).

After calculating these indexes, SOS gives suggestions to the user concerning how to modify some constraints in order increase the quality of the final schedule.

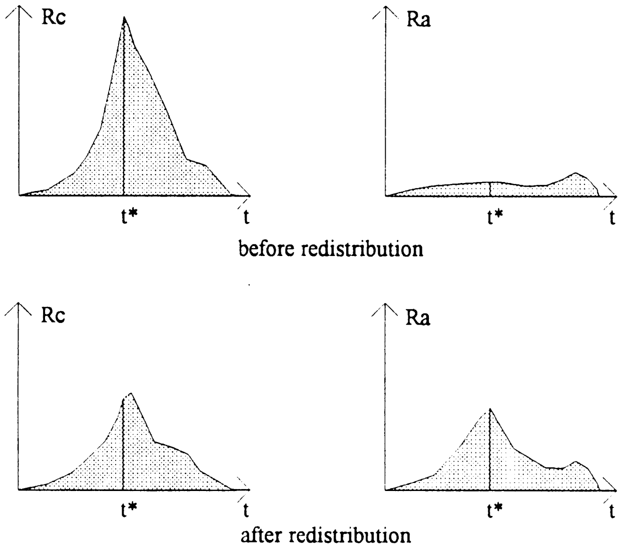

Figure 4. Resource demand redistribution

At present, the pre-analysis consists of three steps:

- indexes are calculated;

- predictive results are displayed to the user, together with suggestions about their modification;
- the user can decide to go back to phase one to re-edit some parts of the problem or to continue in scheduling process.

For example, if resource R_c is highly contended (the presence of a "strong" bottle-neck will surely create problems and will cause a final schedule of poor quality) and if one of activities involved in this bottle-neck can use another resource R_a (which is not contended), the user should modify the problem in order to reduce the aggregate demand of the critical resource, allocating the activity on the non-contended resource. The effects of this reallocation are depicted in figure 4, which show AD values for resources R_a and R_c.

THE SCHEDULING PROCESS

The approach embedded in SOS is not completely related to traditional perspectives in AI-based scheduling: resource-based, order-based and event-based ones. We named our strategy **time-based**: it is similar to the event-based one, but it follows a temporal logic, like resource-based and order-based ones.

The scheduling process always proceeds along time axis. The pure event-based perspective builds the final schedule as a "puzzle"; therefore, it may happen situations in which the not yet scheduled activities do not have enough spaces left and backtracking has to be applied.

The SOS logic, instead, works out the schedule alternating indifferently decisions on the most constrained resources and activities; the scheduler takes each decision only after verifying that all the activities preceeding the selected most constrained object, have enough space to be scheduled. In this way it is possible to avoid the backtracking and to reduce the use of constraints relaxation.

This scheduling policy identifies for each order an ideal line that delimits two zones of the in-progress schedule: the first one contains all the activities already scheduled and the second one all the activities not yet scheduled.

This ideal line has been called "**scheduling wave crest**", due to the fact that this concept can be graphically represented as a line between the two mentioned zones; this line moves from left (zero time) to right (order end) during the scheduling process. The concept of scheduling wave crest is used also to compute schedule in reactive scheduling, by relating it to the happening of unexpected events.

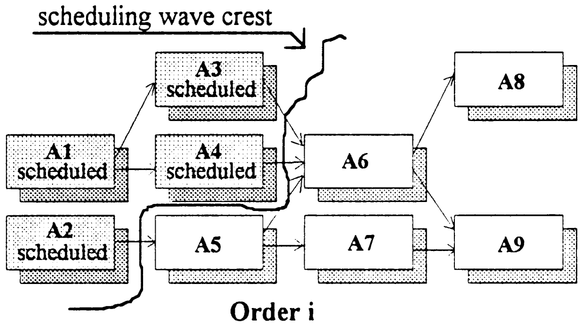

Figure 5. Scheduling wave crest

The explanation of scheduling process requires a preliminary description of constraints propagation and system strategies.

Constraints propagation
linvolves temporal windows and constraints that link two or more activities. Constraints propagation keeps consistent for each activity the start and end time values of the temporal windows during the entire scheduling process.

The decision to schedule activity A_k, belonging to order O_j and requiring resource R_m, causes modifications on the temporal windows of

a) activities belonging to order O_j(intra-order constraints are propagated);

b) activities belonging to different orders that require R_m in the time interval where Ai has been scheduled (propagation of inter-order constraints).

Intra-order propagation involves activities belonging to the same order, and fires from the modification of temporal windows: a scheduling decision fixes the start and end time values and, therefore, causes a propagation towards all the activities temporally linked. Actually, SOS executes an intra-order propagation forward or backward the time axis.

Inter-order propagation, instead, involves activities of different orders that require the same resource during the same time interval. For example, activity

A_k is allocated on resource R_m during time interval I_s. Resource R_m will be unavailable during I_s for all the other activities. Therefore the inter-order propagation effect is the restriction of the temporal window of the other activities. Naturally, the inter-order propagation causes one intra-order propagation for each order involved.

Strategies for the Allocation of activities

After selecting activity A_i to be scheduled on resource R_m, we have to choose among its feasible start times, the one that is contained in an interval of availability for resource R_m and that has the least impact on the allocation of non-scheduled activities.

Therefore we have to examine ST_i, ET_i, D_i and PST_i values. It may occur two different situations: (1) PST_i is contained in ET_i - ST_i interval (Feasible Window - FW_i) or (2) it is not contained.

(1) PSTi is inside the feasible window:

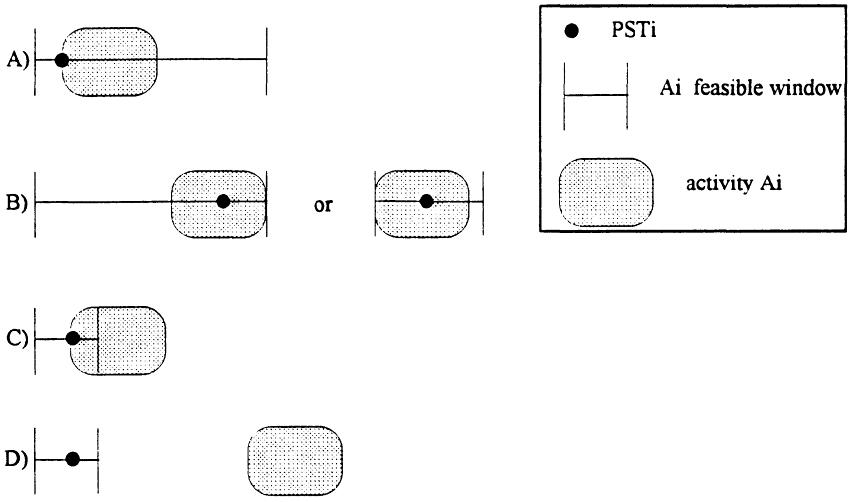

Figure 6. How to schedule activity A_i when PST_i is contained in FW_i.

A) If PST_i+D_i is inside FW_i, then schedule at PST_i, according to R_m availability.

B) If PST_i+D_i is outside FW_i but FW_i can contain A_i ($D_i \leq ET_i$-ST_i), then schedule as near as possible the PST_i value, according to R_m availability.

C) If A_i can not be contained in FW_i, but R_m availability overlaps the window; then schedule at PST_i.

D) If $D_i > ET_i - ST_i$ and resource availability does not overlap the FW_i, then find a suitable time interval of resource availability as near as possible to PST_i, and schedule there.

(2) PSTi is outside the window:

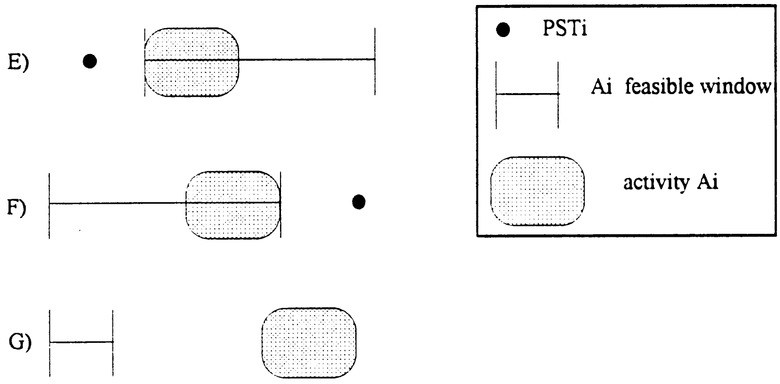

Figure 6. How to schedule activity A_i when PST_i is not contained in FW_i.

E) If $PST_i < ST_i$ and $D_i \leq$, $(ET_i - ST_i)$, then schedule at ST_i time.

F) If $PST_i > ET_i$ and $D_i \leq$, $(ET_i - ST_i)$, then schedule at $(ET_i - D_i)$.

G) A_i cannot be contained in $(ET_i - ST_i)$. Find the first time interval where the resource is available.

Scheduling decisions are taken on the basis information given by the PST_i, which shows the best time where to schedule an activity in order to reduce contentions among activities (value goodness).

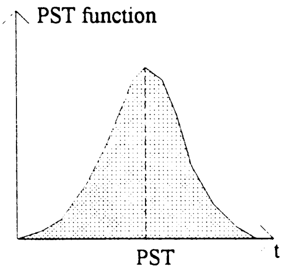

Figure 8. Typical form of PST function

The PST function always has the shape represented in figure 8. If we are not able to schedule an activity at its best start time PST, we have to try to schedule it as near as possible to this value.

The scheduling process

We will distinguish between the terms "allocated" and "scheduled". We will refer to an activity as allocated when its start time is temporary fixed. In this case the scheduler can retract an allocated activity without backtracking. We will refer to an activity as scheduled, when its start time is definitely fixed and constraints have been propagated. Differences between allocating and scheduling, shown in figure 9, mainly establish that a scheduled activity can not become "unscheduled".

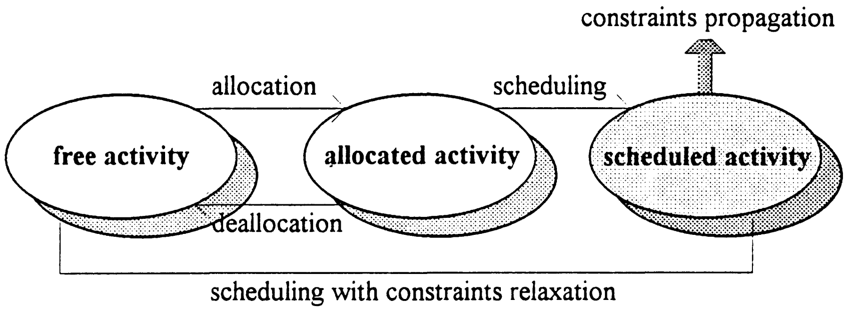

Figure 9. The three possible states of activities

We will use also the following terms:

Current bottle-neck: it is the bottle-neck selected as the focus of attention of scheduling.

Current activity: it is the activity, involved in the current bottle-neck, that has been selected to be scheduled.

Path: it indicates all the possible paths (in a TCCG graph) starting from the scheduling wave crest to the current activity;

R_m **engagement:** it is the set of time intervals where resource R_m is definitely scheduled to an activity.

The approach followed to obtain a solution, points out zones where the resources are most contended by the activities (most constrained strategy). The resolution of these bottle-necks, in our time-based perspective, strictly follows time axis. After a bottle-neck has been individuated, the scheduler processes any not yet scheduled activity involved in this bottle-neck, starting from the most constrained, with an order-based perspective.

The steps of the scheduling process are:

1. Data Base initialization to reset the knowledge base of SOS;

2. The system picks out the first bottle-neck (following time axis) for each resource. If all the bottle-necks have been considered, then go to step 12.

3. Select the current bottle-neck on the basis of the following criteria:

 3a. Select the first bottle-neck following the temporal axis;
 3b. if two or more resources have a contemporaneous bottle-neck, choose the one of them with the greater AD (variable looseness);
 3c. if two or more resources have also the AD value, choose one in a random way.

4. Select all the activities involved in the current bottle-neck;

5. Choose from this set the current activity:

 5a. select the activity that has the greater value of ID index corresponding to the bottle-neck time. The system distinguishes the activities on the basis of this index because it gives global information concerning all orders and is related to the degree of resource contention.
 5b. if the ID index of two or more activities are equal, then chose the one that has the greater V_i index (it is the one with the minimum slack).
 5c. If the set of the activities is empty, then go back to step 2.

6. The system builds in its data base the path for the current activity;

7. The system tries to allocate all the activities included between the wave crest and the current activity, without relaxing constraints and proceeding backward from the current activity to the scheduling wave crest. The instants for allocation are calculated with the same criteria used for scheduling. Each allocation causes a backward intra-order propagation and a propagation towards those activities that belong to the path and use the same resource (this is not properly an inter-order propagation). All the effects produced by the allocation can be quickly undo if at least one inconsistency is detected;

8. If the allocation reaches the scheduling wave crest, without detecting any inconsistency on constraints, the allocating decisions are translated into scheduling decisions, following these steps:

8a. modify the activity state from "allocated" to "scheduled";
8b. update the resource engagement;
8c. execute an inter-order propagation of each decision. Each inter-order propagation causes backward and forward intra-order propagation.
8d. move ahead the scheduling wave crest;

9. Execute a forward intra-order propagation starting from the current activity to the end of its order. Then go back to step 5.

10. If at least one inconsistency is detected, constraints relation is introduced, and the data base is reset to the values existing before the allocation phase. The system begins scheduling one activity at a time, starting from the scheduling wave crest to the current activity following time axis:

10a. move the scheduling wave crest just beyond the selected activity;
10b. schedule the activity at its earliest start time;
10c. update the resource engagement;
10d. modify the state of the activity from "free" to "scheduled";
10e. run inter-order and, consequently, intra-order propagations;
10f. run a forward intra-order propagation for only one step (multiple steps are not necessary because the system is relaxing constraints);

11. Start a forward intra-order propagation from the current activity to the end of its order. Then go to step 5.

12. When all the bottle-necks have been examined, all the activities left are scheduled with an order-based perspective, starting from step 7. This phase do not create new problems, because these activities do not belong to any bottle-neck

RESULTS DISPLAYING AND JUSTIFICATION SYSTEM

A graphic interface displays the scheduling results through two types of Gantt Diagram: the diagram of resources (that shows the total utilization of resources) and the diagram of activities (figure 10). The activities Gantt Diagram displays the temporal collocation of each scheduled activity of the job, giving also all the temporal values that concern the scheduling decisions (the start and end time, etc.). The areas in this diagram are mouse-sensitive; clicking on them user can display information about computed schedule.

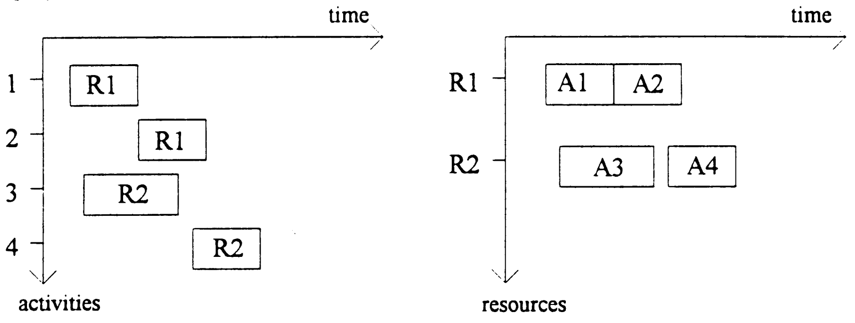

Figure 10. Examples of Gantt Diagrams

The computed solution can be not satisfying for the user (this problem may arise above all when it is necessary to relax many constraints because they are too strict). In this case the user is allowed to change the constraint values to improve the quality of the solution according to his managerial objectives. SOS justifies the relaxing decisions, explaining to the user their causes: following these directives the user can understand which constraints should be modified and how.

The justification system by examining the information gathered during scheduling process and the final schedule, explains to the user the scheduling decisions that decreased the quality of the final schedule. For example, if a manager has set the demand for a resource in a time interval as to exceed its physical capability, it is necessary to relax some temporal constraints to obtain a solution.

IMPLEMENTATION AND CONCLUDING REMARKS

This system has been implemented in ART-IM (Automated Reasoning Tool for Information Management); it is based on rules, schemas, facts and functions (also C functions) and it is a Windows 3.x application that requires at least 6 Mb RAM. Another Windows application, Graphic ToolBook, supplies the interface facilities for input (problem definition) and output (justification system) phases. At the present SOS has been tested on a 486/33 MHz processor, with satisfying results. Scheduling is very quick and the computed schedules are of good quality. We tested the system also with problems that have no solution and we obtained the same satisfying performances.

By now we are developing SOS along two directions: we are improving the mechanisms of pre-analysis , and we are enclosing new classes of preferential constraints in the set of analyzed constraints.

ACKNOWLEDGMENTS

We are much indebited to Paolo Bonetto for his relevant work in implementing SOS. This work has been supported by Mauro Foundation and National Research Council "Special Project on Planning" grants.

REFERENCES

1. Allen, J. *Maintaining Knowledge about Temporal Intervals,* Technical Report TR86, U. of Rochester Department of Computer Science, 1981.

2. Bensana, E., Bel, G. and Dubois S 'OPAL: A multi-knowledge based system for industrial job-shop scheduling' *International Journal of Production Research*, Vol. 26, N.5, pp. 795-819, 1988.

3. Fox, M. 'Observations on the role of constraints in problem solving' *Proceedings of Sixth Canadian Conference on A.I.*, Montreal, 1986.

4. Fox, M., Smith, S.F. and Peng Si Ow, P.S. 'Constructing and Maintaining Detailed Production Plans: Investigations into the Development of knowledge-based Factory Scheduling Systems" *A.I. Magazine*, 1986.

5. Hayes-Roth, B.&F., Rosenschein, S. and Cammarata, S. 'Modeling Planning as an Incremental Opportunistic Process' *Proceedings of the 6th IJCAI*, pp.375-383, Tokyo, Japan, 1979.

6. Keng, N. and Yun, D. 'A Planning/Scheduling Methodology for the Constrained Resource Problem", *proceeding IJCAI*, pp.998-1003, 1989.

7. Paolucci, E., Sem, M. and Patriarca, E. 'A Temporal Predictive Framework for Scheduling Systems" *Working Notes AAAI Spring Symposium Series*, Stanford University, March 1992.

8 Possa, M., Preda, S. and Bonetto, P. 'SOS a Shell for Opportunistic Scheduling', Tesi di Laurea, Politecnico di Milano, 1992.

9.Sadeh, N. and Fox M., 'Preference Propagation in Temporal/Capacity Constraint Graphs', Robotics Institute Carnegie Mellon University, Pittsburgh, 1989, T.R. CMU-CS-88-193.