Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

J.M. Anthonisse, K.M. van Hee, J.K. Lenstra

Resource-constrained project scheduling:
an international exercise in DSS development

Department of Operations Research and System Theory Note OS-N8702 December

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

Resource-Constrained Project Scheduling: an International Exercise in DSS Development

J.M. Anthonisse
Centre for Mathematics and Computer Science (CWI), P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

K.M. van Hee
Department of Mathematics and Computing Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands

J.K. Lenstra
Centre for Mathematics and Computer Science (CWI), P.O. Box 4079, 1009 AB Amsterdam, The Netherlands; Econometric Institute, Erasmus University, P.O. Box 1738, 3000 DR Rotterdam, The Netherlands

The International Institute for Applied Systems Analysis in Laxenburg, Austria, coordinates an international exercise in the development of decision support systems. The participants will independently develop a number of interactive planning systems for resource-constrained project scheduling, in the hope of generating knowledge and experience in the design, analysis and implementation of decision support systems. This report specifies the rules of the exercise.

1980 Mathematics Subject Classification: 90B35, 90C50.
Key Words & Phrases: decision support system, resource-constrained project scheduling, comparative evaluation.

1. Scope and purpose

This report specifies the rules of an exercise in the development of decision support systems. Participants in the exercise are research groups from several countries. Coordination will take place at IIASA, the International Institute for Applied Systems Analysis in Laxenburg, Austria.

The purpose of the exercise is to generate knowledge in methods for designing decision support systems and in the structural and functional properties of such systems. The participants hope to achieve this purpose by means of the independent development of a number of decision support systems for one and the same planning situation and the subsequent comparative evaluation of the resulting systems. The exercise can thus be viewed as a purely empirical research effort.

It should be emphasized again: the exercise has creation of knowledge and experience as its purpose and system development as its means. The systems that are to be developed do not have to be operational in a real-life situation. They are experimental systems for a prototype situation. The planning situation that has been selected, resource-constrained project scheduling, does seem to be eminently suitable in this context. It has three important advantages:

(a) Virtually all operations researchers are to some extent familiar with questions of resource-

Note OS-N8702
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

constrained project scheduling. In the area of artificial intelligence there also is interest in this type of problem.

(b) The planning situation allows for a simple description. More precisely, one can easily define a model that captures all the complexity of practical scheduling situations without being encumbered with a distressing amount of notational formalism.

(c) The planning situation does not allow for a simple solution. That is, there is no efficient algorithm for solving the general resource-constrained project scheduling problem, and there is no hope that such an algorithm will ever be developed. One has to settle for some heuristic mixture of insight, interaction and algorithmics - which is what DSS is all about.

To enable a comparison between the various systems, the participants have to conform to the rules of the exercise. These are detailed below.

Section 2 presents an informal description of the decision situation, a formal specification of instances of the model, and suggestions for extensions of the decision situation. Section 3 mentions a number of functional properties of the systems; some of these are required, others are optional. Section 3 also deals with technical aspects. Section 4 describes the procedures of evaluating and publishing the results of the exercise.

Appendix 1 gives a mathematical description of the primary decision situation. Appendix 2 contains an example of the input and output format.

Potential participants in the exercise are encouraged to contact

Dr. A. Lewandowski
System and Decision Sciences
IIASA
A-2361 Laxenburg
AUSTRIA

or one of the authors of this report for further information. The authors are grateful to A. Lewandowski, I. Stanchev (Sofia), A. Wierzbicki (Warsaw) and a number of potential participants for their comments on drafts of this report.

2. Decision situation

2.1. *Introduction*

The decision situation occurs in the planning department of an organization that runs several projects simultaneously. The planning department communicates both with the operational departments of the organization and, through the sales department, with the customers of the organization.

The sales department supplies the planning department with orders to carry out a project. An order contains a project specification which describes the tasks constituting the project, the relations between the tasks, and the release time, the due date and the deadline of the project. The customers may also stipulate release times, due dates and deadlines for specific tasks.

Within the operational departments, resources (e.g., personnel and equipment) are available to process the tasks during working hours. The processing of a task consists of the application of a function to an object. For example, the packing of a final product is a task, consisting of the application of the function 'packing' to the object 'final product'. Processing times are deterministic but depend upon the resources that are used. For example, it may be that a function can be performed either by resource R_1 with speed 1, or by resource R_2 with speed 2, or by R_1 and R_2 together with speed 3. If a task requires 12 units of this function, then its processing time is either 12 or 6 or 4 time units, depending on the resources that are allocated to it.

The relations between tasks include simple precedence constraints; for example, 'painting' precedes 'packing'. These are generalized by allowing the specification of a lower bound and an upper bound on

the waiting time between the completion of one task and the start of another one. Sometimes, there is no precedence constraint between tasks, but their simultaneous processing is impossible. This is the case if they require the same resource or if they apply to the same object.

At the beginning of each production period, the planning department supplies the operational departments with schedules for that period. A schedule lists the tasks to be processed in that period and specifies for each task its starting time and the resources to be allocated to it. Although the planning department delivers schedules for a single production period, it must develop schedules for several periods in advance. During each planning session all uncompleted tasks of all projects are taken into account.

During the production period, deviations from the schedules occur. Actual processing times may differ from those which were used to develop the schedules. Sometimes, an operational department fails to perform a task. In general, such tasks must be included in the schedule for a subsequent production period.

Near the end of the production period, which is close to the beginning of the next one, the planning department receives a full report on the state of the production process. This report lists the completed tasks and the tasks that were scheduled for this period but will not be started in this period. Moreover, each resource which will be active at the beginning of the next production period is reported, together with the task then allocated to it and the time still to be spent on that task. The planning department has only a short period of time to produce the schedules for the next production period. A DSS should provide the means to quickly develop good schedules.

In this primary problem situation, the quality of the schedules is derived from the service to the customers, who compare the actual completion times of tasks and projects with their due dates and deadlines. Other quality measures are discussed in Section 2.5.

A planning session will generally start with the inspection and processing of data from the sales department and from the operational departments. From these data, the parameter values of a decision problem are derived. This problem type will now be described more precisely.

2.2. *Verbal description of problem type*

A set of tasks is to be processed by a set of resources. For each task, there is a release time and a deadline, which define a time interval in which the task must be processed. Once a task is started it must be completed without interruption.

For any two tasks, there are a lower bound and an upper bound on the length of the time period between the completion of one task and the start of the other. Simple precedence constraints are obtained by setting the lower bound to zero and the upper bound to an appropriately large number.

A function may be performed by various combinations of resources, each with its own speed. In general, each function has a class of feasible resource sets, and the processing time of a task depends on the feasible resource set that is chosen to perform the function it requires. The processing time is the amount of work (i.e., the number of units of the function) required by the task divided by the speed of the feasible resource set.

No resource can be allocated to two tasks at the same time. If a set of resources is allocated to a task, then each of its constituent resources is occupied by that task from its starting time until its completion time. For each resource there is a set of time intervals during which the resource is available.

The above description suffices to define feasible schedules. To measure the quality of a schedule, we need at least one criterion. The due date of a project or a task is a point in time at which it should ideally be completed. The length of the time period between the completion of a task and its due date is called earliness (if completion occurs before the due date) or tardiness (if completion occurs thereafter); these concepts are relevant for 'just in time' planning strategies. Earliness and tardiness of a task are multiplied by the earliness weight and tardiness weight of that task, respectively. There are now four obvious overall criteria: maximum earliness (i.e., the maximum weighted earliness of any task), maximum tardiness, total earliness (i.e., the sum of the weighted earlinesses over all tasks), and total tardiness. Each weighted sum of these four is an overall criterion. All these weights are crucial parameters. They can be used to specify task priorities and to choose between minmax and minsum criteria.

The input format allows the specification of various sets of earliness weights and tardiness weights for

the tasks and various sets of overall weights. A DSS should provide support for the minimization of a single criterion as well as for the minimization of multiple criteria.

2.3. *Comments on problem type*

Before defining the data structure of the problem, let us comment on its generality, its complexity, and its limitations. First, any model representing this situation covers a rich variety of problem types, including network planning and all the favorites from deterministic machine scheduling theory. Secondly, optimization may seem like a difficult task, but even answering the simple question if there exists a feasible schedule or not is, in general, far from trivial. Finally, many aspects that would make the scheduling situation more realistic are not included. To mention a few:

- preemption, or task splitting, is not allowed;
- simultaneous processing of more than one task by the same resource, or resource splitting, is not allowed;
- change-over times, which can occur when two tasks are performed consecutively by the same resource, are not considered;
- there are no random effects: the model is completely deterministic.

In spite of these limitations, we feel that the planning situation as sketched above is already general and complex enough. No matter how many bells and whistles we append, anyone involved in practical production planning can come up with further additions, each of them perfectly reasonable from a real-world point of view. For the exercise, however, we do not need a perfect picture of a real planning situation, but a suitable prototype.

2.4. *Data structure of problem instance*

The purpose of this section is the specification of the structure of the data that define a problem instance and a solution in terms of a *relational data model.* In Section 3.2 the format of a file to represent these data will be specified.

We distinguish four sets of tables for the specification of *projects*, *resources*, *criteria*, and *schedules*, respectively.

Each table is defined by the name of the table and the list of the names of its attributes. The attributes that together constitute a key for the table will be indicated. References between tables will also will be indicated. Some attributes are indicated as optional, which means that they may have the value nil. With each table the integrity requirements are discussed.

2.4.1. *Projects.* The names of the tables which specify the projects are *projects*, *tasks*, and *precedence*, respectively.

The table *projects* has the following attributes:

project identifier	(key)
release time	(optional)
due date	(optional)
deadline	(optional)

All tasks of the project should be processed in the time interval as defined by the release time and the deadline of the project. Ideally, the project should be completed at its due date, which lies in the time interval [release time, deadline].

If the release time is nil, then it can be reset to a sufficiently early time, which might be derived from the release times of the tasks or from the opening times of the resources. Similarly, if the deadline is nil, it can be reset to a sufficiently late time, which might be derived from the deadlines of the tasks or from the closing times of the resources. If the due date is nil, then the project has no due date but one or more tasks may have their own due dates.

The table *tasks* has the following attributes:

project identifier	(key, refers to projects)
task identifier	(key)
function identifier	(refers to functions)
function capacity requirement	
release time	(optional)
due date	(optional)
deadline	(optional)
object identifier	(optional)

The function capacity requirement is the amount of work required by the task. The processing time of a task is obtained by dividing this amount by the speed of the resource set which is allocated to it. The task should be processed in the time interval defined by its release time and its deadline. Ideally, the task should be completed at its due date, which lies in the time interval [release time, deadline]. The release time and deadline should be in the time interval defined by the release time and deadline of the project. If the release time or deadline are nil, then they can be reset to sufficiently early or late times, respectively, which might be derived from the release time and deadline of the project or from the opening times and closing times of the resources. If the due date is nil, then there is no due date for this task. If the object identifier is nil, then there is no object for this task, or, equivalently, the object is unique for this task.

The table *precedence* has the following attributes:

project identifier	(key, refers to projects)
task1 identifier	(key, refers to tasks)
task2 identifier	(key, refers to tasks)
minimum waiting time	(optional)
maximum waiting time	(optional)

Precedence constraints occur only between tasks of the same project. The waiting time is the length of the time interval between the completion time of task1 and the starting time of task2. If the minimum waiting time is nil, then it is reset to zero. If the maximum waiting time is nil, then there is no upper bound on the waiting time. The minimum and maximum waiting time may be negative. If the table does not contain a precedence constraint between two tasks, then these tasks can be processed in any order, without bounds on the waiting time.

2.4.2. *Resources*. The names of the tables which specify the resources are *resources*, *availabilities*, *resource sets*, and *functions*, respectively.

The table *resources* has the following attributes:

resource identifier	(key)
kind identifier	(optional)
department identifier	(optional)

If the kind identifier is nil, then the kind of the resource is apparently left unspecified. If the department identifier is nil, then the resource does not belong to a specific department. This table is included to facilitate the grouping of resources according to their kind (e.g., personnel or equipment) and the operational departments of the organization. This may be used in the presentation of data and results and even to analyze the problem and to develop a solution strategy.

The table *availabilities* has the following attributes:

resource identifier	(key, refers to resources)
opening time	(key)
closing time	

The resource is available from its opening time until its closing time. If several time windows are specified for the same resource, then they should be disjoint. The minimum of the opening times of the resources is the beginning of the next production period.

The table *resource sets* has the following attributes:

resource set identifier	(key)
resource identifier	(key, refers to resources)

The resource is a member of the set. A resource may be a member of several sets.

The table *functions* has the following attributes:

function identifier	(key)
resource set identifier	(key, refers to resource sets)
speed	

2.4.3. *Criteria.* The names of the tables which specify the criteria are *task criteria*, *project criteria*, and *overall criteria*, respectively.

The table *task criteria* has the following attributes:

task weights identifier	(key)
project identifier	(key, refers to projects)
task identifier	(key, refers to tasks)
earliness weight	(optional)
tardiness weight	(optional)

If a weight is nil, it is reset to zero. If a task with a due date does not occur in this table, then both weights are zero. If a task occurs in this table but has no due date, then the weights are ignored.

The table *project criteria* has the following attributes:

project weights identifier	(key)
project identifier	(key, refers to projects)
earliness weight	(optional)
tardiness weight	(optional)

If a weight is nil, it is reset to zero. If a project with a due date does not occur in this table, then both weights are zero. If a project occurs in this table but has no due date, then the weights are ignored.

The table *overall criteria* has the following attributes:

criteria identifier	(key)
project weights identifier	(key, refers to project criteria, optional)
task weights identifier	(key, refers to task criteria, optional)
maximum earliness weight	(optional)
total earliness weight	(optional)
maximum tardiness weight	(optional)
total tardiness weight	(optional)

The project weights identifier and the task weights identifier cannot both have the value nil. If a weight is

nil, it is reset to zero. If all four weights are zero, then the project criteria and the task criteria as referred to by the project weights identifier and the task weights identifier respectively, are considered as multiple criteria. Otherwise, the four weights define a single criterion.

2.4.4. *Schedules.* The name of the table which specifies one or more schedules is *schedules.* It has the following attributes:

schedule identifier	(key)
criteria identifier	(key, refers to overall criteria)
project identifier	(key, refers to projects)
task identifier	(key, refers to tasks)
resource set identifier	(key, refers to resource sets)
starting time	
completion time	(optional)

2.5. *Extensions*

The problem situation as described above is oriented towards the optimal meeting of due dates, where optimality is defined with the help of one or more criteria. Instead of or alongside these criteria, other criteria may be of interest. Resources might be available at a price and the planner might wish to minimize total costs or to use the tardinesses of projects and the costs as simultaneous criteria. Another criterion to measure the quality of a schedule is its robustness against perturbations of the speeds of resources. The participants are encouraged to supplement their problem instances with criteria of their own choice. This should, however, not disable their systems to assist in solving the primary problem type.

3. Decision support system

3.1. *Functional requirements and suggestions*

The minimal functional requirements of a system are its ability to read any problem instance as input and to write a schedule as output. It would be useful if it can also read schedules (that, for example, serve as initial solutions) and write problem instances.

Between reading and writing, there is arithmetic as well as interaction with the user of the system. It is up to the participants to design and implement functions which enable the user to mix his insight with algorithmics.

We indicate some functions as suggestions for the participants.

Consistency checking. The system might help the planner to detect inconsistencies in the input data.

Updating of data. During the planning session the planner might wish to reset weights or other data in order to guide the solution process. Needless to say, the original problem instance must be used to assess the feasibility and the quality of the final schedule.

Manual scheduling. The planner might wish to develop a schedule from scratch or to modify a schedule.

Automatic scheduling. The system might be able to generate or modify a schedule, without human interference.

Feasibility relaxation. The system might allow the planner to treat certain constraints (e.g., release times and deadlines) as soft constraints.

Inspection. The system could provide different views of a problem instance or a schedule.

3.2. *Technical specifications*

For exercises in man-machine interaction, the use of two types of resources has to be indicated. As to human effort, about one man-year seems reasonable. As to machinery, the choice is also left to the participants. However, during the comparative evaluation each system should be available at IIASA. The participants should then either use a machine provided by IIASA (i.e., an IBM PC/XT or AT with a MicroSoft mouse and CGA or EGA card, or a compatible configuration) or bring their own equipment. Furthermore, the machine should be able to use 5¼ inch, DS, DD disks.

We now define the format of ASCII files for the exchange of data among the participants. A problem instance and a schedule consist of the tables as defined in Section 2.4 in that order. We assume that each problem instance will be stored as a separate file. Another file will contain one or more schedules. On the file each table consists of a headline, followed by a number of lines with data and closed by an endline.

The *headline* consists of an asterisk in the first position, immediately followed by the name of the table.

Each *dataline* contains the values of the attributes, in the order as specified in Section 2.4. The first position of these lines is left blank. Subsequent attributes are separated by a comma; the last attribute is followed by a semicolon. All attributes have a fixed length: ten positions for identifiers (left justified) and seven positions for numerical data (right justified). If all positions are blank, then the attribute has the value nil. Numerical data must be given in scientific format (Pascal format).

The *endline* consists of an asterisk in the first position.

Data may be interspersed with *comments*. An exclamation point indicates that the remainder of the line contains comments only.

Extensions of the primary problem can be specified with the help of additional tables, which should be appended to the tables defined here.

A program to test files for conformity with the formats will be available to the participants.

4. Evaluation and publication

4.1. *Comparative evaluation*

Progress will be discussed during meetings in the spring and the fall of 1988. The final evaluation will be at IIASA in the spring of 1989. The evaluation includes the testing of systems on problems from other participants. The size of problems to be run is restricted by the available ardware. Each participant should provide a functional description of his system to the meeting.

4.2. *Publication*

It is envisaged that the results of the exercise will be published in book form under the auspices of IIASA. The book should contain a description of the design and implementation of each of the contributed systems and a report of their comparative evaluation. It could also include papers on original methodological or algorithmic aspects of the systems. The copyright of a software product developed in the context of the exercise remains entirely with the researcher or group of researchers or institute responsible for developing it.

Appendix 1. Mathematical definition of problem type

We define an optimization model with a single criterion function. The participants in the exercise may choose to use multiple criteria of the types defined here. Moreover, they may use additional criteria of other types and they may treat constraints of the problem as soft constraints.

The model does not include the concept of a project. The release times, due dates and deadlines of projects can be modelled with the help of appropriately defined dummy tasks. The availability of resources can be modelled by means of an appropriate task for each time interval during which a resource is unavailable. The model does not include the concept of objects to which functions are applied. These objects are considered as resources. The object associated with a task is a member of each feasible resource set for that task.

Data. There is a set $\mathbf{R}$ of resources and there is a set $\mathbf{T}$ of tasks. For each task $T \in \mathbf{T}$, there is

- a release time $a(T)$ and a deadline $b(T)$,
- a class $\mathbf{F}(T)$ of feasible resource sets, with $\mathbf{F}(T) \subset 2^{\mathbf{R}}$ (the class of all subsets of $\mathbf{R}$),
- a due date $d(T)$, an earliness weight $v(T)$ and a tardiness weight $w(T)$.

For each ordered pair of tasks $(T,U) \in \mathbf{T}\times\mathbf{T}$, there are limits $\alpha(T,U)$ and $\beta(T,U)$ on waiting time. For each task $T \in \mathbf{T}$ and each of its feasible resource sets $F \in \mathbf{F}(T)$, there is a processing time $p(T,F)$. Finally, there are four weights $v_{\max}$, $w_{\max}$, v_{sum}, w_{sum}.

Schedule. A schedule is a pair (F,S) of functions on $\mathbf{T}$. In schedule (F,S) task $T \in \mathbf{T}$ is processed by a feasible resource set $F(T) \in \mathbf{F}(T)$ and has a starting time $S(T)$. We define the completion time of task $T \in \mathbf{T}$ by $C(T) = S(T)+p(T,F(T))$.

Feasibility. A schedule (F,S) is feasible if it satisfies
- release dates and deadlines:

  $\forall T \in \mathbf{T}$: $a(T) \leqslant S(T)$, $C(T) \leqslant b(T)$;

- limits on waiting time:

  $\forall T,U \in \mathbf{T}$: $\alpha(T,U) \leqslant S(U)-C(T) \leqslant \beta(T,U)$;

- resource capacities:

  $\forall T,U \in \mathbf{T}$: $F(T) \cap F(U) \neq \emptyset \Rightarrow \big(C(T) \leqslant S(U)$ or $C(U) \leqslant S(T)\big)$.

Optimality. A schedule (F,S) defines for each task $T \in \mathbf{T}$:
- a weighted earliness $V(T) = v(T)\cdot\max\{0,d(T)-C(T)\}$,
- a weighted tardiness $W(T) = w(T)\cdot\max\{0,C(T)-d(T)\}$.

These are aggregated in four criteria:
- the maximum earliness $V_{\max} = \max_{T\in\mathbf{T}}\{V(T)\}$,
- the maximum tardiness $W_{\max} = \max_{T\in\mathbf{T}}\{W(T)\}$,
- the total earliness $V_{\text{sum}} = \sum_{T\in\mathbf{T}} V(T)$,
- the total tardiness $W_{\text{sum}} = \sum_{T\in\mathbf{T}} W(T)$.

The overall optimality criterion is given by

$$Z = v_{\max} V_{\max} + w_{\max} W_{\max} + v_{\text{sum}} V_{\text{sum}} + w_{\text{sum}} W_{\text{sum}}.$$

The objective is to find a feasible schedule that minimizes Z.

APPENDIX 2. EXAMPLE OF PROBLEM INSTANCE AND SCHEDULE

```
! example of problem instance
! in this example there are two projects,
! each consists of the printing and binding of a book
*projects
!project_id,release,    due,   dead;
 vol1      ,       ,    100,    130;
 vol2      ,       ,    125,    140;
*
*tasks
!project_id,task_id   ,function  , capreq,release,    due,   dead,object    ;
 vol1      ,print     ,print     ,    500,     19,     70,       ,          ;
 vol1      ,bind      ,bind      ,    600,       ,      ,       ,          ;
 vol2      ,print     ,print     ,    250,     40,      ,       ,          ;
 vol2      ,bind      ,bind      ,    400,       ,      ,       ,          ;
*
! printing precedes binding
*precedence
!project_id,task1_id  ,task2_id  ,minwait,maxwait;
 vol1      ,print     ,bind      ,     10,     30;
 vol2      ,print     ,bind      ,     10,     40;
*
! the shop has two presses but a single operator
! there are two men for binding the books
*resources
!res_id    ,kind_id   ,dept_id   ;
 press1    ,eq        ,pr        ;
 press2    ,eq        ,pr        ;
 Smith     ,p         ,pr        ;
 Wright    ,p         ,bn        ;
 Fisher    ,p         ,bn        ;
*
*availabilities
!res_id    ,   open,  close;
 press1    ,     20,   1000;
 press2    ,      1,   1000;
 Smith     ,     17,   1000;
 Wright    ,    125,   1000;
 Fisher    ,      0,     30;
 Fisher    ,     40,   1000;
*
! if a press is run it requires the constant attention of an operator
! Wright and Fisher can work simultaneously on the same task
*resource sets
!set_id    ,res_id    ;
 run1      ,press1    ;
 run1      ,Smith     ;
 run2      ,press2    ;
 run2      ,Smith     ;
 Wr        ,Wright    ;
 Fi        ,Fisher    ;
 Wr_Fi     ,Wright    ;
 Wr_Fi     ,Fisher    ;
*
! continued on next page
```

```
! continued from previous page
*functions
!func_id    ,set_id     ,  speed;
 print      ,run1       ,     25;
 print      ,run2       ,      6;
 bind       ,Wr         ,     30;
 bind       ,Fi         ,     35;
 bind       ,Wr_Fi      ,     70;
*
! the earliness of the printing of vol1 is of interest
*taskcriteria
!t_w_id     ,project_id,task_id    , earl_w, tard_w;
 epv1       ,vol1       ,print     ,      1,       ;
*
! the tardinesses of the two projects are to be minimized
*projectcriteria
!p_w_id     ,project_id, earl_w, tard_w;
 tard       ,vol1       ,       ,      1;
 tard       ,vol2       ,       ,      1;
*
!crit_id    ,p_w_id     ,t_w_id     , max_ew, tot_ew, max_tw, tot_tw;
 crit1      ,           ,epv1       ,      1,       ,       ,       ;
 crit2      ,tard       ,           ,       ,       ,       ,      1;
 crit3      ,tard       ,           ,       ,       ,       ,       ;
*


! example of schedule
*schedule
!sched_id   ,crit_id    ,project_id,task_id    ,set_id     ,  start,    compl;
 sch1       ,crit2      ,vol1      ,print      ,run2       ,     17,      101;
 sch1       ,crit2      ,vol1      ,bind       ,Fi         ,    115,      133;
 sch1       ,crit2      ,vol2      ,print      ,run1       ,    101,      111;
 sch1       ,crit2      ,vol2      ,bind       ,Wr         ,    125,      139;
*
```