

RESCON

User's Manual

Table of contents



1.	Working with projects.....	1
a.	Loading a project	3
b.	Zooming features	3
c.	Editing a loaded project	3
d.	Saving projects.....	4
e.	Project metrics & statistics.....	4
2.	Calculating schedules.....	4
a.	Working with the Gantt chart and resource profile windows	4
b.	Early/late start schedules.....	5
c.	Resource feasible schedules.....	5
i.	Constructive heuristics.....	6
ii.	Tabu search	6
iii.	Exact branch-and-bound	6
iv.	Comparative analysis: the algorithm performance summary window	7
v.	Robust schedules.....	7
vi.	External schedule generation code (DLL)	7
3.	Robust scheduling.....	7
a.	Robust scheduling data	8
i.	Constraints on the robust scheduling data	8
ii.	Reading the robust scheduling data from a file	8
iii.	Writing the robust scheduling data to a file	9
iv.	Modifying the robust scheduling data.....	9
b.	The STC heuristic	10
4.	Writing your own scheduling DLLs	11

1. Working with projects

a. Loading a project

Projects can be loaded in one of two ways: it can either be read in from a file, or it can be built up from scratch by the user.

To read a project from a file, choose **File → Open...** and select an .rcp file. The rcp file format is the format used in PSPLIB¹. Warning: if the selected file contains logical errors (e.g. the number of indicated successors is not equal to actual the number of successors supplied in the file), RESCON might crash without warning.

To build up a project from scratch, select **File → New**. Enter the desired number of renewable resources and their availabilities. This will create a project with a dummy start and dummy end activity (not visible in the network window), without any other non-dummy activities. Non-dummy activities can then be added one at the time using the “new activity” button () from the toolbar. In the “New activity” dialog, you can select any number of predecessors for the new activity just by clicking on the activities in the list box. Whenever a project network is loaded, you must first close the active file (using either  or **File → Close**) before you can load a different project.

b. Zooming features

If a project network is hard to read, the built-in zooming feature may alleviate the problem. With the project network window active, use the keys “+” and “-” or the mouse wheel to zoom in or out in the project network.

c. Editing a loaded project

A loaded project can be modified in almost every thinkable way:

- **add/remove successors of an activity:** successors can be added or removed by double-clicking on an activity. This will cause the “activity properties” dialog to appear, in which successors can be added or removed simply by clicking on the corresponding activities in the listbox.
- **change activity durations or resource requirements:** again, double-click on an activity. To change the resource requirement for a specific resource type, select the resource type from the listbox and change the requirement.
- **add/remove activities:** adding activities was already explained in section 1.a. Removing an activity can be achieved by first selecting an activity (by clicking on it): the selected activity will be colored red. Press the “delete” key to delete the selected activity.
- **change the resource availabilities:** changing resource availabilities can be done by double-clicking anywhere in the whitespace of the project network window (be careful not to click on an activity). Resource availabilities can not be lowered in a way that would render an activity non-executable.

¹ <http://129.187.106.231/psplib/>

d. Saving projects

Projects can be saved in the aforementioned rcp format using the menu item **File** → **Export as rcp...**

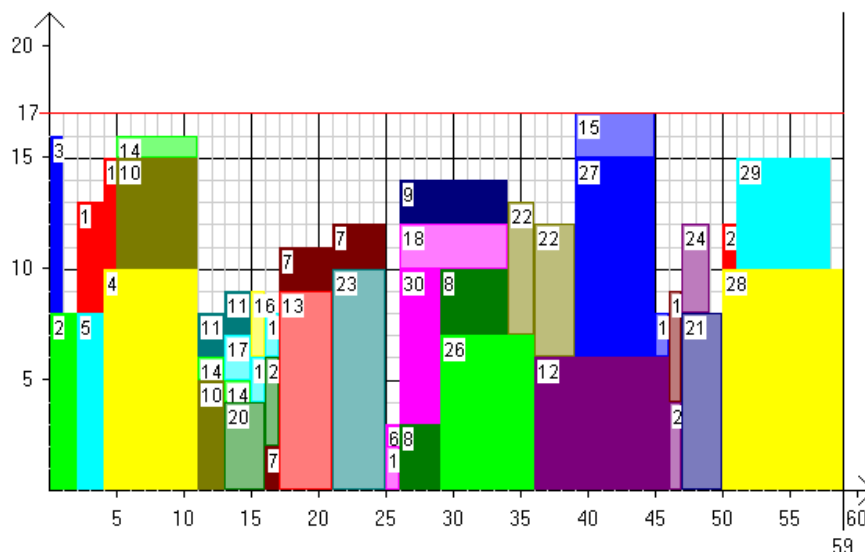
e. Project metrics & statistics

With the project network window active, several metrics and statistics (e.g. resource strengths) can be calculated by clicking the menu item **Action** → **Show project statistics**.

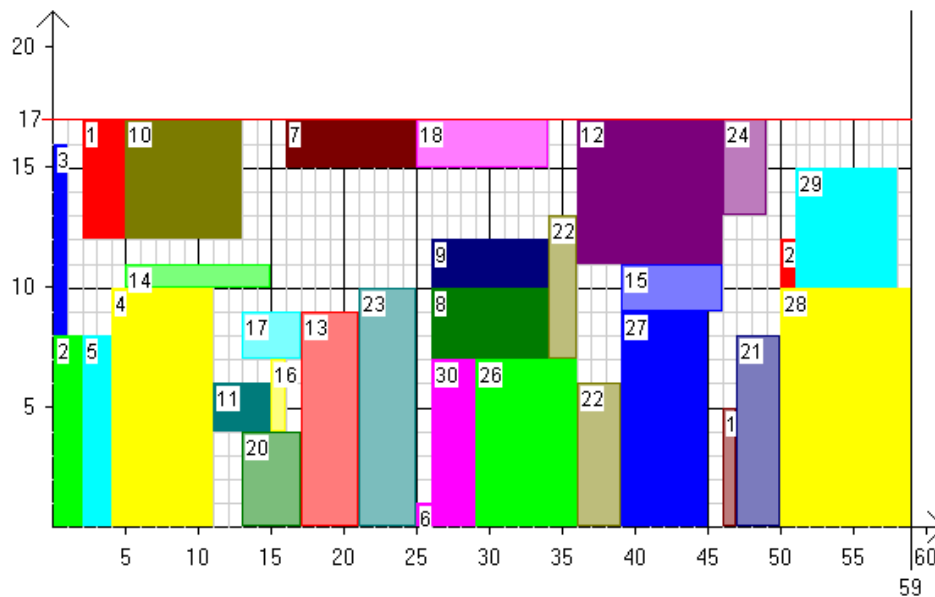
2. Calculating schedules


a. Working with the Gantt chart and resource profile windows

When a schedule has been calculated (see sections **2.b** and further), resource profiles and a vertical Gantt chart can be drawn. By default, whenever a new schedule is calculated, a resource profile will be shown for every resource type. (TIP: use the tile button (🧩) for a clear overview of all resource profiles). Resource profiles can be drawn in one of two modes. Click on the “skyline profile” button (🏞️) to display the profile in a skyline kind of fashion. This view has the advantage that peaks in the resource consumption are immediately visible. The disadvantage is that activities are not displayed as contiguous rectangles:




If clear activity visibility is the priority, the user can opt for the “contiguous activity” resource profile mode by clicking this button: 🏠. With this option selected, RESCON will try to display every activity as a contiguous block. For the above example, this will result in a resource profile looking like this:



A Gantt chart of the current schedule can be drawn upon request. To use this feature, click the “Gantt chart” button () . The gray bars extending from the side of some of the activities indicate slack values. If an early (or late) start schedule is loaded (see next section), the slack values indicate the time difference between the earliest and the latest starting time of each activity. If a resource feasible schedule is loaded, the slack values indicate the amount an activity can be right-shifted without delaying the starting time of one of its successors, and without violating the resource constraints.

When certain windows are closed by the user, they can be re-opened by selecting the appropriate menu item from the [Window](#) menu. A schedule is considered loaded as long as the Gantt chart window or one of the resource profile windows is still open. When a schedule is loaded, all windows corresponding to it (i.e. the resource profiles and the Gantt chart) can be re-opened by clicking the appropriate menu item. When the network window is closed, it can be re-opened *only if* any of the aforementioned schedule windows is still open. If none of these windows is open, the file is considered closed and the project must be loaded anew from the [File](#) → [Open...](#) menu item.


b. Early/late start schedules

By ignoring the resource constraints, early and late start schedules can be calculated by the traditional backward/forward calculations. To calculate early or late start schedules, select the appropriate menu item from the [Algorithm](#) menu. Then, click on the “calculate schedule” button () .


c. Resource feasible schedules

RESCON features a diversity of algorithms to calculate resource feasible schedules. Furthermore, the user has the opportunity to summarize the results of all algorithms in a single window. These features are further explained in the next sections.

i. Constructive heuristics


The conceptually simplest procedure to obtain a resource feasible schedule is probably list scheduling. RESCON features forward, backward and bidirectional scheduling procedures that can be complemented with either the serial or the parallel schedule generation scheme. These procedures all require a priority list of the project activities as input. Eight well-known priority rules are embedded in RESCON (amongst other things LFT, SPT, MINSLK, ...). This yields a total of 48 possible ways to generate a feasible schedule using list scheduling. To generate such a schedule, click on the menu item [Algorithm → Constructive heuristic](#) and select the appropriate settings from the dialog box. Then, click on the “calculate schedule” button (). *Remark: when you change the algorithm type while one of the Gantt or resource profile windows open, the schedule will be recalculated immediately.*

ii. Tabu search

In order to strike a balance between computation time and solution quality, you can choose to use the built-in (rudimentary) tabu search procedure. To do so, select the menu item [Algorithm → Tabu search](#) and enter the desired computation time in the dialog box. Click the “calculate schedule” button () to start the algorithm. While the algorithm is running, no other operations can be performed: the user has to wait for the algorithm to finish. The progress of the algorithm can be monitored from the status bar.

iii. Exact branch-and-bound


For relatively simple project networks, the user may choose to calculate the optimal schedule (with respect to makespan performance). A variant of the well-known branch-and-bound algorithm developed by Demeulemeester & Herroelen² has been implemented. The implemented version supports projects up to 500 activities under the constraint that one can not form a set of more than 32 activities such that any pair of activities in the set is precedence unrelated. In other words, large projects are supported only if the level of inherent parallelism is limited. This is the only real constraint imposed. A word of caution is appropriate here: if the project is sufficiently large or “difficult”, the procedure may take very long to complete, and may consume a lot of memory. Projects with up to twenty activities should be solvable without any problem. For larger projects, the computation time and the memory requirements will depend on the inherent complexity of the problem. The decreased efficiency of the procedure when compared to the published version of the algorithm lies in reasons related to the greater generality of the implemented variant and the way MFC applications work.

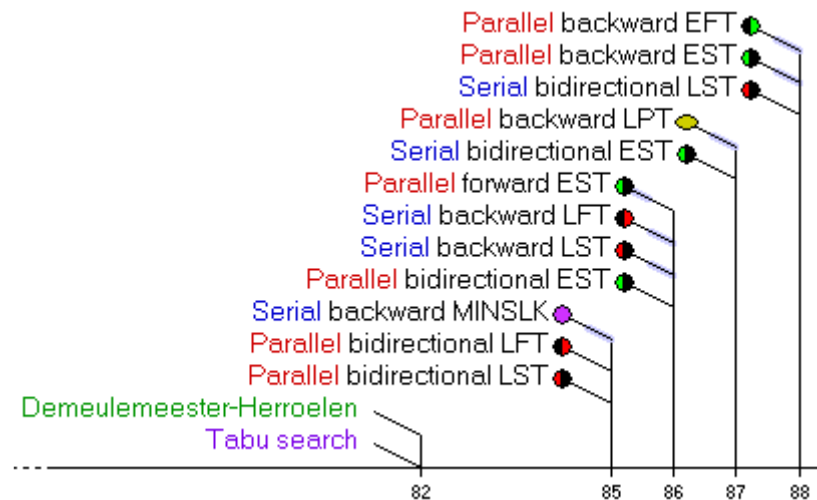
To calculate an optimal schedule, select the menu item [Algorithm → Demeulemeester-Herroelen](#) and click the “calculate schedule” button (). To some extent, the progress of the algorithm can be monitored

² Demeulemeester, E., and Herroelen, W., 1992, A branch-and-bound procedure for the multiple resource-constrained project scheduling problem, *Management Science* 38(12), pp 1803-1818.

from the status bar. The progress percentage is calculated on the basis of the number of visited first and second level nodes.

iv. Comparative analysis: the algorithm performance summary window

For educational purposes, it may be interesting to look at the comparative performance of the different scheduling algorithms described in the above sections. These results can be summarized in the “algorithm performance summary” window. To activate this window, select the menu item [Window → Show algorithm performance summary window](#) or click the appropriate button () from the toolbar. A dialog will be displayed that enables you to check (or uncheck) two time-consuming algorithms from the set, namely tabu search and optimal branch-and-bound. After submitting your preferences, you will get a window that looks like this:



(only a part of the actual window is displayed here)

The obtained makespan from the selected algorithms is depicted on the horizontal axis.

v. Robust schedules

Robust schedules can be generated by selecting the menu item [Algorithm → STC](#). For more information, see Section 3 of this manual.

vi. External schedule generation code (DLL)

To use external schedule generation code, select the menu item [Algorithm → User defined \(DLL\)](#). For more information on this, see Section 4 of this manual.

3. Robust scheduling

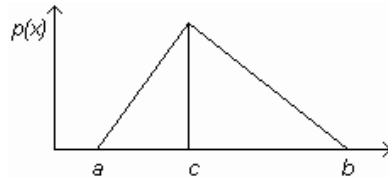
Up till now, we have ignored the robust scheduling component of the software. Two topics will be discussed in this section: the robust scheduling data and the STC heuristic, a heuristic for generating robust schedules.

a. Robust scheduling data

Besides the deterministic data of the project (i.e., the data that is available in files in the rcp format), some additional data is stored internally concerning the project activities. More specifically, for every activity, the software keeps track of the activity's *inflexibility weight* and of the parameters of the *triangular duration distribution* of the activity. When a project in the rcp format is read in, the software automatically generates a random activity duration distribution per activity, along with a random activity inflexibility weight. This is also done for any activity that is added manually to the project, through the RESCON user interface.

i. Constraints on the robust scheduling data

The parameters of the triangular duration distributions are subject to a number of constraints. Three parameters are saved: the lowest possible activity duration (a), the most likely activity duration (c), and the highest possible activity duration (b). These parameters determine the probability density function of the triangular distribution. Evidently, the following



constraint must be satisfied on these parameters: $a \leq c \leq b$. In addition to this constraint, we also require that $a \leq d_i \leq b$ with d_i the deterministic activity duration. Finally, we require that $b \leq \max(10, 2 * d_i)$.

If the user changes d_i into a value that violates these constraints, the parameters of the triangular duration distribution are updated silently so that the conditions remain satisfied.

Besides the triangular distribution, every activity (except for the dummy start) has an inflexibility weight. This positive number may vary between 0 and 100. The dummy end activity also has an inflexibility weight, reflecting the costs (per time unit) incurred when the planned project finish time is not met.

ii. Reading the robust scheduling data from a file

Given an active project, the user may choose to read the robust scheduling data from a file. This data will replace all robust scheduling data currently associated with the project. Robust scheduling data files must be in ascii (txt) format, and must be formatted according to this example:

```
0 0 0 0 0
1 3 1 5 8
2 5 1 6 8
3 2 1 4 6
4 1 1 3 5
5 5 1 3 4
...
```

The file must contain one line per activity (dummies are included), and every line contains the following information:

```
[activity number] [inflex. weight] [param. a]
[param. c] [param. b]
```


If these numbers do not satisfy one of the constraints mentioned in (ii), the numbers will be adjusted so that the constraints are satisfied and the user will be notified of the adjustments.

To read robust scheduling data from a file, select **File → Read robust scheduling data** and select the appropriate file. You will get a message if the information was successfully read in, and the project network window will show the **[modified]** flag.

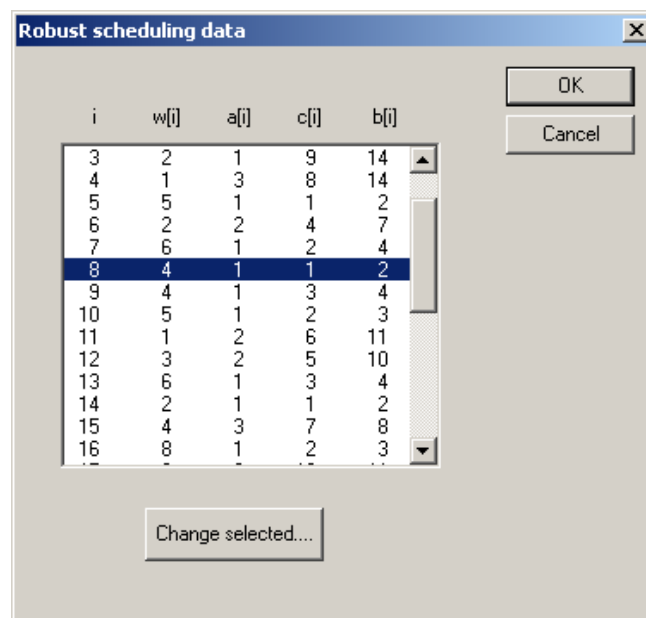
iii. Writing the robust scheduling data to a file

The currently loaded robust scheduling data can be written to a file in the right format, using the menu item **File → Save robust scheduling data**.

iv. Modifying the robust scheduling data

Besides reading new data from a file, there are two additional ways to change the robust scheduling data of the loaded project. The first way involves double-clicking on an activity. In the activity properties window that pops up, the data can be changed. If the changes are confirmed with the **OK** button, the **[modified]** flag will be set.


This method does not allow the inflexibility weight of the dummy end activity to be changed, because that activity is never shown in the project network window. Therefore, we have foreseen another method of changing the robust scheduling data. With the project network window active, select the menu item **Window -> Show robust scheduling data**. The following dialog will pop up:

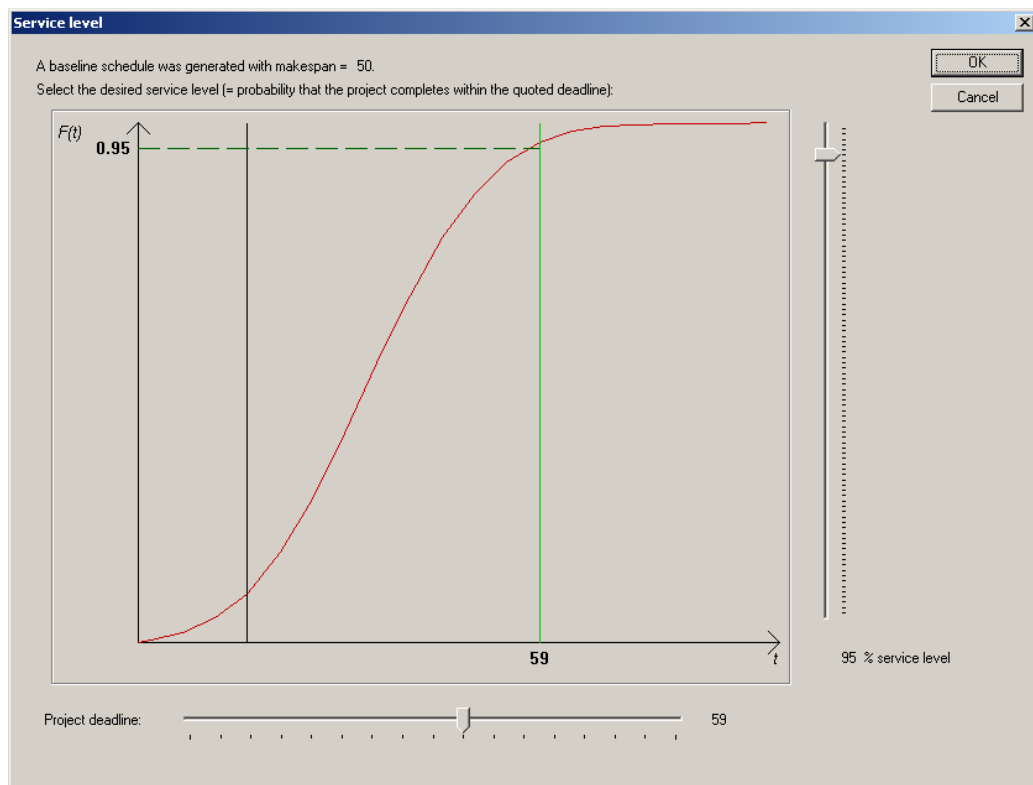


i	w[i]	a[i]	c[i]	b[i]
3	2	1	9	14
4	1	3	8	14
5	5	1	1	2
6	2	2	4	7
7	6	1	2	4
8	4	1	1	2
9	4	1	3	4
10	5	1	2	3
11	1	2	6	11
12	3	2	5	10
13	6	1	3	4
14	2	1	1	2
15	4	3	7	8
16	8	1	2	3

This dialog gives a summary of the robust scheduling data. For every activity, the activity number, the inflexibility weight and the parameters a , c and b are shown. The user can then select any activity (including the dummy end activity) and subsequently press the “change selected” button to change the corresponding robust scheduling data. Confirmation with **OK** will again trigger the **[modified]** flag to be shown in the project network window.

b. The STC heuristic

The STC heuristic³ generates a robust schedule using the information concerning activity duration distributions and activity inflexibility weights. To do so, select **Algorithm → STC...** and choose an algorithm for the generation of the initial schedule from the dialog that shows. After this choice has been made, nothing will happen (unless there is already a Gantt chart window or a resource profile open). To continue, press the “calculate schedule” button (). Doing so will trigger the selected initial schedule generation algorithm. Depending on your choice, the calculation of the initial schedule may take some time. A next dialog will be shown:



In this dialog, you will be notified of the makespan obtained during the generation of the initial schedule. With this makespan corresponds the black vertical line in the cumulative project duration distribution. A Monte-Carlo simulation was performed to calculate the cumulative project duration distribution using the initial schedule as a starting point for project execution. The curve links project deadlines to service levels (i.e. the probability that the project will actually complete within the quoted deadline). The user should use this curve to pick an appropriate deadline using the slider bars. That deadline will then be used by the STC procedure to generate a schedule with a planned finish time equal to that deadline. Select a deadline, then press **OK** to generate the buffered schedule. As always, all resource profiles corresponding with the

³ Van de Vonder, S., Demeulemeester, E. & Herroelen, W. (2008). Proactive heuristic procedures for robust project scheduling: An experimental analysis. *European Journal of Operational Research* 189 (3): 723-733.

calculated schedule will be shown. The buffers, however, can be more clearly observed in the corresponding vertical Gantt chart.

4. Writing your own scheduling DLLs

The software supports the integration of external schedule generation code into the software. This can be done by means of a DLL. This works as follows. When RESCON is started, the software will check for the presence of a dll called [dlltest.dll](#) in the same folder as the RESCON main executable. If such a dll has been found, the dll will be linked dynamically to the software, and the menu item [Algorithm → User defined \(dll\)](#) will be enabled. If not, the menu item will be grayed out. The source code of the dll itself must contain the implementation of the function [getSchedule\(...\)](#). The dll source code may look as follows:

```
#ifndef _MYDLL_INCLUDED
#define _MYDLL_INCLUDED

#define DLL_EXPORT __declspec(dllexport)
#define DLL_IMPORT __declspec(dllimport)

extern "C"
{
    DLL_EXPORT void getSchedule(int act, int res, int *avail, int *dur, int **req,
                                int *nrpr, int **pred, int **su, int *nrsu, int **_bestSchedule);
}

#endif

extern "C"
{
    DLL_EXPORT void getSchedule(int act, int res, int *avail, int *dur, int **req,
                                int *nrpr, int **pred, int **su, int *nrsu, int **_bestSchedule){

        int *mySchedule = *_bestSchedule;
        // write your schedule generation code here
        // before returning, fill in the activity start times in mySchedule
    }
}
```

The *signature* of the function `getSchedule` is fixed and should be exactly the same in your implementation as in the code shown above. The function takes 10 arguments:

- **int act**: this is the number of activities in the project, including dummies. For instance, in a project with 10 non-dummy activity, `act` will be equal to 12. Activity indices in the code range from **0** (dummy start) to (**act – 1**) (dummy end).
- **int res**: the number of renewable resource types. Resource type indices range from **0** to (**res-1**).
- **int *avail**: for every renewable resource type, `avail[i]` equals the per-period availability, $i = 0, \dots, res-1$.
- **int *dur**: for every activity, `dur[i]` equals the deterministic activity duration, $i = 0, \dots, act-1$. We always have `dur[0] = dur[act-1] = 0`.

- `int **req: req[i][k]` equals the per-period requirement of activity `i` of resource type `k`.
- `int *nrpr: nrpr[i]` equals the number of predecessors of activity `i`.
- `int **pred: pred[i][p]` is the index of the p^{th} predecessor of activity `i`. `i = 0, ..., act-1`; `p = 0, ..., nrpr[i]-1`.
- `int *nrsu`: the number of successors of every activity `i`. See `nrpr`.
- `int **su`: the indices of the successors of every activity `i`. See `pred`.
- `int **_bestSchedule`: this is a *pointer* to an array that should contain your schedule. Fill in this array before returning from the function. For `i = 0, ..., act-1`, you should do something along the lines of
`(*_bestSchedule)[i] = [your calculated start time of activity i];`

Some other important remarks:

- You may assume that, IF activity `j` is a successor of activity `i`, THEN the activity indices `i` and `j` will ALWAYS satisfy the constraint `j > i`. Be careful!! The activity indices WILL NOT correspond to the indices shown on the screen in the activity network window. The only certainty is that activity 0 is the dummy start and activity `act-1` is the dummy end.
- You should not `delete` any of the arrays passed to the function. The RESCON software will take care of that. You should, however, make sure not to cause any memory leaks of your own.
- Fully functional DLL example source code is made available on the RESCON website. A corresponding Visual Studio 6.0 project is available as well.
- The DLL support has only been tested in Visual Studio 6.0. That is, we can guarantee that DLLs written in Visual C++ 6.0 are compatible with the software. Later versions of Visual Studio have not been tested in that regard.

