

# SCHEDULING TOOLBOX FIRST PREVIEW

*Michal Kutil*

Department of Control Engineering  
Faculty of Electrical Engineering, Czech Technical University

**Abstract.** Scheduling theory has been a popular discipline for a last couple of years. However, there is no tool, which can be used for a complex scheduling algorithms design and validation. Creation of this tool is our goal and its first preview is described in this paper. The tool is written in Matlab object oriented programming language and it is used in Matlab environment as a toolbox. Main objects are Task, TaskSet and Problem. Object Task is a data structure including all parameters of the task as process time, release date, deadline etc. Objects of a type Task can be grouped into a set of tasks and other related information as precedence constrains can be added. Object Problem is a small structure describing classification of deterministic scheduling problems in Graham and Błażewicz notation. These objects are used as a kernel providing general functions and graphical interface, making the toolbox easily extensible by other scheduling algorithms.

## INTRODUCTION

Scheduling algorithms were known many years ago, but currently there are no good tools used to work with those algorithms. Existing tools are usually developed to solve just one specific problem, not for general problem solution. Presented toolbox is intended to be used for solving those general scheduling problems. The first major part of the toolbox is used to interpret, present and save data from scheduling toolbox. The second part implements basic scheduling algorithms. Data from scheduling problems can be divided into four groups: tasks, resources, precedence constrains and optimality criterions.

## TASK

Task is a basic term in scheduling problems, which describes any unit of work that is scheduled and executed by the system. The task is described by the following parameters:

<b>Name</b> (Name)	label of the task
<b>Processing time</b> (ProcTime)	is the time necessary to the processor for executing the task without interruption
<b>Release time</b> (ReleaseTime)	is the time at which a task becomes ready for execution
<b>Deadline</b> (Deadline)	specifies a time limit by which the task has to be completed, otherwise the scheduling is assumed to fail
<b>Due date</b> (DueDate)	specifies a time limit by which the task should be completed, otherwise the criterion function is charged by penalty
<b>Weight</b> (Weight)	expresses the priority of the task with respect to other tasks
<b>Processor</b> (Processor)	specifies dedicated processor at which the task must be executed

The task is represented by the object data structure with the name *task* in Matlab. This object is created by the command with the following syntax rule:

```
t1 = task([Name,]ProcTime[,ReleaseTime[,Deadline[,DueDate  
[,Weight[,Processor]]]])
```

Command *task* is a constructor for object of type *task* whose output is stored into a variable (in the syntax rule above it is the variable *t1*). Properties contained inside the square brackets are optional. Creating the objects of the type *task* in matlab are shown on figure 1.

```
>> t1 = task(5)
Task ""
  Processing time: 5
  Release time:   0
>> t2 = task('task2',5,3,12)
Task "task2"
  Processing time: 5
  Release time:   3
  Deadline:      12
>> t3 = task('task3',2,6,18,15,2,2)
Task "task3"
  Processing time: 2
  Release time:   6
  Deadline:      18
  Due date:      15
  Weight:        2
  Processor:     2
```

Fig. 1. Creating a task objects

## A SET OF TASKS

Objects of the type *task* can be grouped into a set of tasks. A set of tasks is an object of the type *taskset* and can be created by the command *taskset*. Syntax for this command is as follows:

```
T = taskset(tasks[,prec])
```

where variable *tasks* is an array of objects of the type *task* and *prec* is a matrix containing precedence constrains between tasks. If there are not any precedence constrains between the tasks, we can use a shorter entry for creating a set of tasks (Fig. 2 – first line). The command *plot* can be used to draw a set of tasks in the Gantt chart (Fig. 3).

```
>> T = [t1 t2 t3]
Set of 3 tasks
>> T=taskset(T,[0 1 1; 0 0 1; 0 0 0])
Set of 3 tasks
  There are precedence constraints
>> plot(T)
```

Fig. 2. Creating a set of tasks and adding precedence constrains

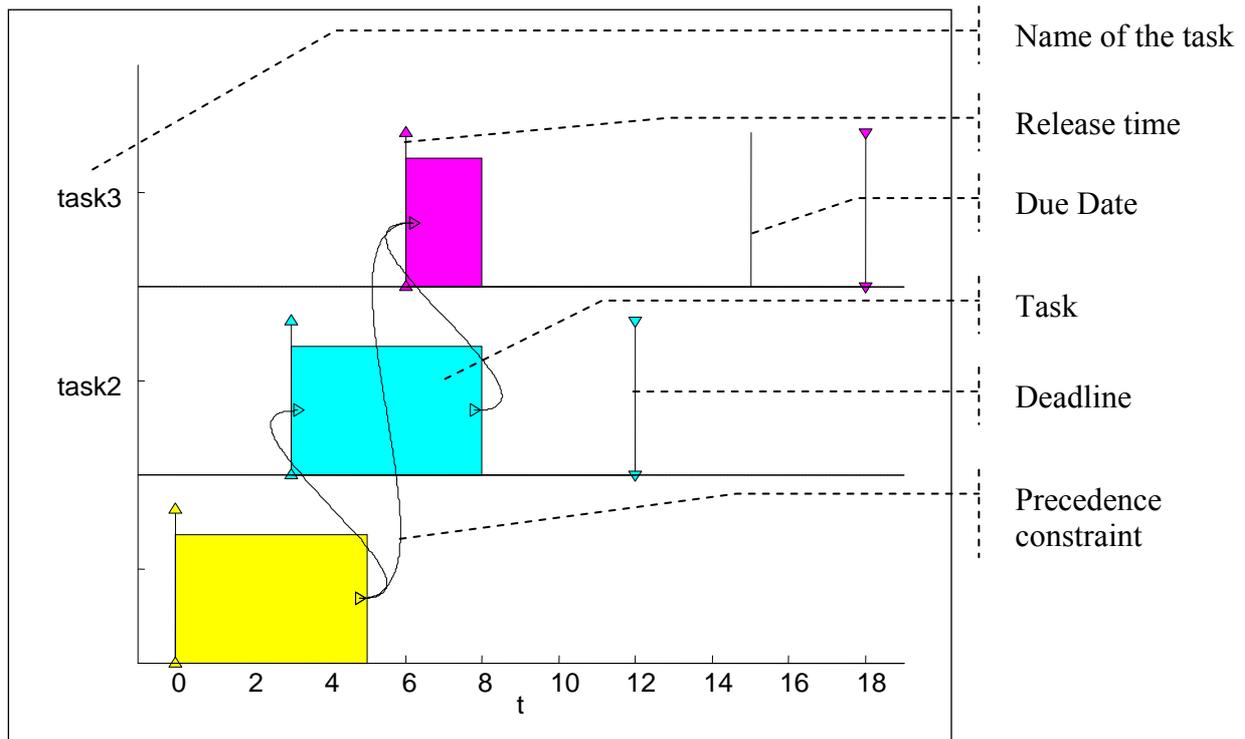


Fig. 3 Gantt chart for a set of scheduled tasks

## PROBLEM

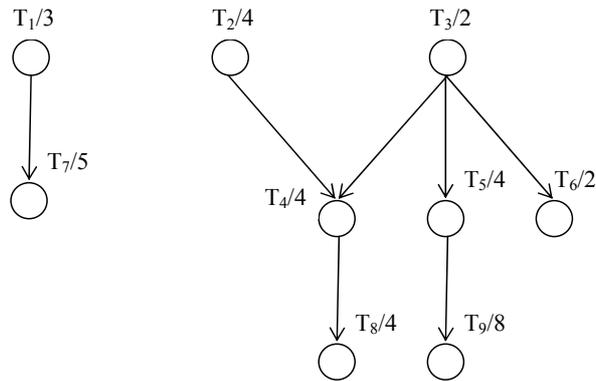
The object *problem* is a small structure describing the classification of deterministic scheduling problems in the notation proposed by Graham et al. [1] and Błażewicz et al. [2]. An example of its usage is shown in the following code.

```
>> p = problem('P|prec|Cmax')
P|prec|Cmax
```

This notation consists of three parts ( $\alpha | \beta | \gamma$ ). The first part (alpha) describes the processor environment, the second part (beta) describes the task characteristics of the scheduling problem as the precedence constraints, or the release time. The last part (gamma) denotes an optimality criterion. The command *is* is used to ask whether any notation includes specific description.

## CASE STUDY

In this chapter a solution of the P|prec|Cmax problem [3] is shown. There are nine tasks with precedence constraints, see Fig. 4. Processing time of each respective task is written after its name.



**Fig. 4 submission of scheduling problem**

Data inscription and solution of this problem by the scheduling toolbox is shown on Fig. 5.

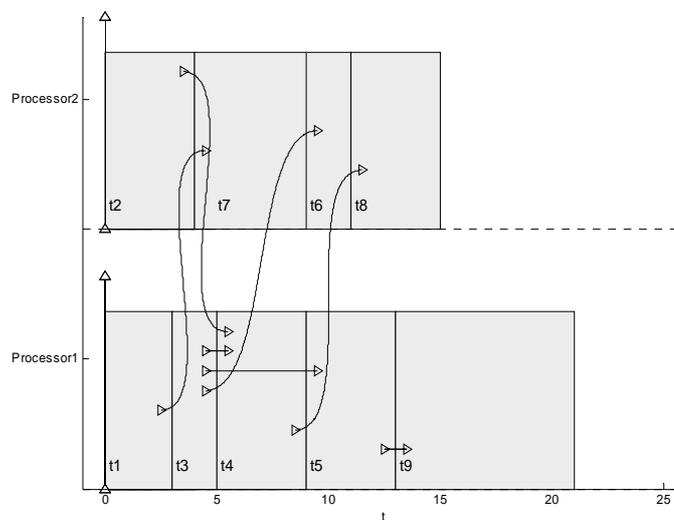
```

>> t1 = task('t1',3);
>> t2 = task('t2',4);
>> t3 = task('t3',2);
>> t4 = task('t4',4);
>> t5 = task('t5',4);
>> t6 = task('t6',2);
>> t7 = task('t7',5);
>> t8 = task('t8',4);
>> t9 = task('t9',8);
>> prec = full(sparse(...
    [1,2,3,3,3,4,5],[7,4,4,5,6,8,9],[1,1,1,1,1,1,1],9,9));
>> T = taskset([t1 t2 t3 t4 t5 t6 t7 t8 t9],prec);
>> p = problem('P|prec|Cmax');
>> Tsolution = listsch(T,p,2);
>> plot(Tsolution,'color',0)

```

**Fig. 5 Solution of the scheduling problem**

The command *listsch* is built-in function of the scheduling toolbox. This function computes the schedule by the List scheduling algorithm [3]. The code of this function is included in the appendix. The final schedule is shown on Fig. 6. This chart was drawn by the command *plot*.



**Fig. 6 Final schedule**

## CONCLUSION AND FUTUREWORK

Commands from the scheduling toolbox mentioned above are just a part of all the commands. These commands are necessary to show the basic idea and function principle. This article is just the first preview and is presented to open the discussion about this topic. We would like to connect our scheduling toolbox with Matlab Web Server and to prepare web interface for this toolbox. This interface helps to include your own problem into the scheduling toolbox and solve this problem online via internet. All the commands built in the tool are listed in the appendix.

## LITERATURE

- [1] R. L. Graham, E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling theory: a survey, *Ann. Discrete Math.* 5, 1979, 287-326.
- [2] J. Błażewicz, J. K. Lenstra, A. H. G. Rinnooy Kan, Scheduling subject to resource constraints: classification and complexity, *Discrete App. Math.* 5, 1983. 11-24.
- [3] J. Błażewicz, K. H. Ecker, E. Pesch, G. Schmidt, J. Węglarz, *Scheduling Computer and Manufacturing Process*. 2<sup>nd</sup> printing. Springer, 2001. ISBN 3-540-41931-4
- [4] G. C. Butazo, *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*. Kluwer Academic Publishers, 1997. ISBN 0-7923-9994-3

## CONTACT

**Michal Kutil** (kutilm@fel.cvut.cz)  
Department of Control Engineering  
Faculty of Electrical Engineering, Czech Technical University in Prague  
Karlovo náměstí 13, 121 35 Praha, Czech Republic

## APPENDIX

### List Of Commands In Scheduling Toolbox

General objects of toolbox.

task	create object of task
taskset	create set of tasks
problem	define scheduling problem

Scheduling algorithms

edf	earliest deadline first
listsch	list scheduling

Tasks

add_scht	add schedule to the task
get	access values of task properties
plot	draw the task with properties
set	set/modify properties of task

Set of tasks (taskset)

add_schedule	add schedule to taskset
count	returns amount of taskt inside the set of tasks
get	access values of taskset properties
plot	draw the tasks
set	set/modify properties of taskset

Problem

is	check if property of standard notation is set
----	---

## List Scheduling

```
% Inicialization
pr = get(taskset,'prec'); % precedens constrains
n = count(taskset); % number of tasks
nonassign = ones(1,n);

start = zeros(1,n);
processor = zeros(1,n);
lenght = get(taskset,'proctime');

si = zeros(1,m);

% List Scheduling
for j = 1:n
    [sk,pk] = min(si);
    timecondition = max(repmat(((~nonassign).*(lenght+ ...
        start))',1,n).*pr) <= sk;
    % predecessor is finished or
    % isn't started or is without predecessor

    withoutpr = ~(nonassign*pr) ;
    % without predecessor or preddecesor is scheduled

    nawp = find(nonassign.*withoutpr.*timecondition);
    % non-assigned which are without predecessor or
    % predecessor is finished

    nawp = nawp(1); % First from list

    % add to schedule
    start(nawp) = sk;
    processor(nawp) = pk;
    si(pk) = si(pk) + lenght(nawp); %sk = sk + pj
    nonassign(nawp) = 0; % Take out from list
end

add_schedule(taskset,'List Scheduling', ...
    start,lenght,processor);
```