DOI: 10.15439/2015F389
ACSIS, Vol. 5

# Constructive heuristics for technology-driven Resource Constrained Scheduling Problem

Paweł B. Myszkowski
Wrocław University of Technology, Wrocław, Poland
Email: pawel.myszkowski@pwr.edu.pl

Michał Przewoźniczek
Wrocław University of Technology MP2 company, Wrocław, Poland
Email: michal.przewozniczek@pwr.edu.pl

Marek Skowroński
Wrocław University of Technology, Wrocław, Poland
Email: m.e.skowronski@pwr.edu.pl

*Abstract*—**In this paper, we define a new practical technology-driven Resource Constrained Scheduling Problem (t-RCPSP). We propose three approaches, applying constructive heuristics to tackle effectively the practical application of RCPSP. In the RCPSP formulation, the constraints are defined to design the tasks in the spaces constructed by non- and renewable resources, without violating the precedence relationships and technologies in real world problem that exists in Plastic and Rubber Processing company. The difficulty of t-RCPSP is NP-hard and we proposed three constructive specialized methods: duration based heuristics (DBH), locally optimal resource usage PEC and NEH heuristic adaptation. The paper presents results of computational experiments that show the effectiveness of the proposed approaches.**

## I. Introduction

The automated computer-aided scheduling in real world application has a tremendous impact on the enterprise. Production schedule building process by human needs a lot of time (long hours), what increases costs and strongly depends on the human condition (costly mistakes). Moreover, the automated scheduling process requires less time (only seconds), is faultless and can be run anytime, e.g. to reschedule in the case of break-down production. In most cases, schedule generated by computer is more efficient than schedule built by the human.

In this paper, an automated scheduling problem practical application in Plastic and Rubber Processing Industry is investigated. Mainly, there is a set of injection molding machines, specialized devices, set of (sub)products and ingredients. Such renewable (e.g. machines, devices) and non-renewable (product's ingredients) resources should be assigned to client requests (tasks) to get near optimal usage in the production process. The major element of automated scheduling system is schedule builder. If solution is to be useful in practice, schedule builder should give the (sub)optimal production schedule in reasonable time: less than 1 minute is acceptable.

It is widely known that the automated computer-aided scheduling in real world application may reduce human work. However, our specific domain requirements make complicated application of classical algorithms. We propose three types of RCPSP solving methods: duration based heuristics (DBH) based on the greedy algorithm, classical NEH adaptation and method (PEC locally optimal resources usage driven.

The proposed technology-driven t-RCPSP can be generalized to RCPSP, which in literature is presented as NP-hard [1] and there are no exact algorithms to solve it in reasonable computation time. Some researches recommend heuristics [7][8]Error: Reference source not foundError: Reference source not found as fast and quite effective RCPSP solving tools.

To get schedule near optimal some metaheuristic approaches are recommended, e.g. Simulated Annealing [2], Tabu Search [12][15], Genetic Algorithms [18][22], Evolutionary Algorithms [5]Error: Reference source not found (hybrids EA [21]). Also, some swarm intelligence methods can be successfully applied to RCPSP, like Ant Colony Optimisation [4][9][10][11], hybrid ACO [13], Particle Swarm Optimization [23] or Bee Colony Algorithms [25].

The rest of the paper is organized as follows. Section 2 presents general RCPSP problem statement and specific domain requirements; technology-driven t-RCPSP model is proposed. Section 3 describes details of three proposed heuristics. Experiments of developed methods in a given dataset are presented in section 4. Finally, section 5 presents summary of gained results and gives some possible further research directions.

## II. Problem statement

In this section, the main elements of classical RCPSP are presented. In a real world problem, such RCPSP model can be useless. The main reason is that, in practical application, to realize the client's request machines and devices can use several configurations of ingredients that may cause entirely different task duration. Thus, problem that we met in MP2 company enforced us to extend RCPSP by several elements. Proposed technology-driven RCPSP (*t-RCPSP*) model in details is presented below.

### A. *Short description of technology-driven RCPSP*

In classical RCPSP Error: Reference source not found each task is described by duration, start and finish time. Tasks are non-preemptive, which means that preemption is not allowed. Each task can be lined to other one in timeline. We use discrete time measure - timeslots.

In the presented t-RCPSP application (schema is presented on Figure 1), we need to run several tasks, which are non-preemptive. Each task has its execution deadline and duration time that depends on used resources. To apply task and produce required product(s) some resources are used: machines, devices, materials and subproducts.

Resources are renewable (machines and devices) and non-renewable: materials (such plastic, paints and other ingredients) and subproducts. As some products are composed of other products, there is relation start-finish between tasks that produce needed subproducts of the given task. Some resources (machines and devices) are dedicated, what means that can be assigned only one activity at a given time [6].

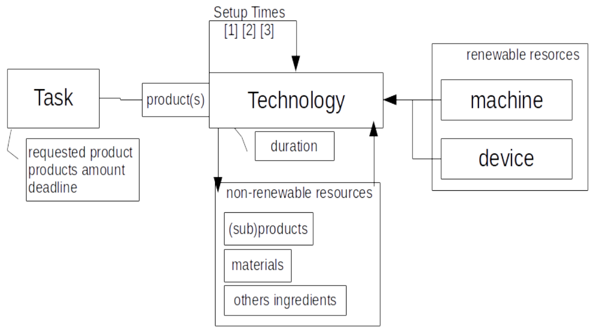

Fig 1. Schema of t-RCPSP

In t-RCPSP specific set of constraints that should be satisfied is defined. The feasible schedule satisfies all constraints defined as follows:

C1. Each task is applied only on one proper machine using specialized device,
C2. Each machine and device can be used only once in selected timeslot,
C3. Each device can cooperate with machine in various way, using other configuration of ingredients,
C4. Each task requires a given amount of ingredients: materials and subproducts,
C5. The task that produces subproducts must be finished before task requiring it,
C6. Each task has defined deadline and number of products,
C7. Each task has assigned duration time that depends on number of required products and used machine and device,
C8. There are 4 types of machine setup times that depend on two adjacent tasks:

- no operation – if two tasks produce the same products,
- start (15 timeslots duration) two tasks produce the same products and machine has been stopped,
- rinse (30 timeslots) two tasks use the same device and machine but provide other products,
- full refitting (120 timeslots) to clean machine and change device.

All devices are specialized to provide a given type of products using the machine, materials or/and subproducts. The device can be applied only to selected machines, and its effectiveness is connected with machine and configuration of ingredients. The main goal of t-RCPSP is to generate feasible schedule (according to C1-C8 constraints) to minimize its duration – makespan, calculated as the difference between first task start and end of the last task in the final schedule. The minor criteria is to reduce the average latency of schedule execution given as the difference between each task end time and its defined deadline.

This problem is NP-hard [1][3][6] and overconstrained. There are no effective algorithms therefore we propose use some heuristics to solve it in acceptable time. The extra constraint required by MP2 company is time limit, i.e. solving method execution cannot exceed 1 minute of CPU computational time of reference machine.

## B. t-RSPSP - formulation

The feasible schedule ($S$) consists $j=1,..,J$ tasks and each task is defined as a tuple:

$$J{:}=<\{request\_products\ [amounts]...\},\ sj,\ dtj,\ ddj> \tag{1}$$

where $ddj$ defines task execution deadline, $sj$ means timeslot to start task in the discrete time period; However $dtj$ value strongly depends on used resources: machine, device and materials. To link such aspects model we defined technology $t=1,..,T$ as follows:

$$T{:}=<M,\ D,\ \{resources\ I\ [amount]\},\ \{products\ P\ amount\},\ dtt> \tag{2}$$

where $dtt$ value determines the task execution time using given set of resources. To apply technology to produce products ($P$) it uses renewable resource ($m=1,..,M$ machines and $d=1,..,D$ devices) and some product ingredients ($I$) as non-renewable resources: $MA=1,..,i$ materials and other resources $R=1,..,r$, including subproducts.

Various technologies can produce the same product using other resources and give other execution time. Such technology definition as an abstract layer makes possible to link the same resources in another way. Set of technologies describes the effectiveness of model and makes optimization simpler. The primary optimization goals are defined as follows:

$$\min MAKESPAN(S) = \max(sj+ddj) - \min(sj) \tag{3}$$

Such formula gives information about the total schedule $S$ execution time, calculated as the differences between the last task's finish and start of the first task. It should be minimized to make schedule execution possible shorter. Another measure that gives quality of given schedule $S$ is the average latency defined as follows:

$$\min AVG\_LATENCY(S) = \tag{4}$$

$$\frac{1}{k}\sum_{1..k}\left\{\begin{matrix}0\, if\ sj+dtj<ddj\\ else\ ddj-sj+dtj\end{matrix}\right\}$$

Such measure gives the averaged value of how late each task is due to its defined deadline. It should be minimized to finish each task before its deadline and possible to avoid the delay (and potential financial penalties).

## III. Proposed methods

Each of three proposed methods: NEH adaptation, duration based heuristic (DBH) and resource optimal usage PEC are based on some observations and motivations. Moreover, methods differ not only in implementation but they also use various parameters. Proposed heuristics use sorting deadline criteria of tasks. We defined three basic criteria: ascending (tasks with earlier deadline have priority), descending (the opposite situation) and random order. We decided to implement the random task order to get reference to the other two. In this section, details of proposed methods are presented.

### *A. Duration based heuristic – DBH*

The main motivation of DBH is to build the shortest schedule using adaptation of classical greedy algorithm based on rule heuristics [19]. The DBH pseudocode is presented on Pseudocode 1. DBH heuristic works on all unassigned tasks and proposes first possible timeslot and uses the shortest technology to execute it.

The DBH heuristic asks model for set of tasks that can be preformed in selected timeslot (line 6). List of tasks is sorted by criteria (randomly, ascending or descending deadlines) to get one task (line 9). Then the technology with the shortest execution time is given to apply in given timeslot (line 10). If all model constraints are satisfied task is assigned in the schedule (line 11) and removed from list of unassigned tasks (line 12). If there no tasks that can run, the model takes next timeslot (line 13).

### *B. Local optimal resource usage heuristic – PEC*

In DBH heuristic technology is selected that gives the shortest time of task realization. Such strategy is optimal locally because doesn't take into consideration optimal renewable resource usage. In PEC heuristic (see Pseudocode 2) such aspect is included as some local search method. As DBH only assigns the first task, PEC heuristic tries to assign the larger number of tasks in given timeslot (line 11-17). All analyzed tasks are unassigned for schedule (line 19-23). Only the best task sequence for given timeslot is selected and all included tasks are assigned to final schedule (line 26-29).

To reduce the PEC computation complexity some limits are introduced – the *size* PEC parameter defines number of tasks that are analyzed in one sequence. As *size* parameter equals to 1 PEC heuristic works as DBH, the greater value needs much more CPU working time but returns production schedule more efficient.

The PEC heuristic is a type of compromise between semi-blind greedy DBH and brute force method that analyzes all possible permutations to get the (local) optimal schedule. The *size* parameter gives a range of above compromise to get possible better schedule than build by DBH.

### *C. NEH2 as NEH heuristic adaptation*

Results of experiments with PEC and DBH heuristic showed that tasks sequence for effectiveness of algorithms have big impact on the final schedule. Such observation encourage us to find algorithm which can optimize this aspect. The classical NEH (Nawaz, Enscore, Ham) [14] algorithm is considered as one of the most effective method of minimizing the makespan for Permutation Flowshop Scheduling Problem. The main goal in original NEH is to find the optimal sequence of operations to get more optimal

Pseudocode 1. DBH pseudocode

```
    procedure DurationBasedHeuristic ( SORT_CRITERIA )

1    UT = Tasks              // unsigned tasks
2    TS = 0;                 // timestamp
3    RUT_TS = {}             // sequence of ready to run unassigned tasks
4    do
5     assigned=false
6     RUT_TS = getApplicableTasks (UT, TS)
7     if ( |RUT_TS| > 0 )
8       RUT_TS:= SORT( RUT_TS, SORT_CRITERIA );
9       Task = RUT.getFirstTask();
10      Tech = getShortestDurationApplicableTechnology( Task, TS )
11      assigned = schedule.assign (Task, Tech, TS)
12    if (assigned==true) UT = UT / Task
13    else TS++
14   while ( |UT|>1 )
```

Pseudocode 2. PEC pseudocode

```
     procedure PEC ( SIZE, SORT_CRITERIA )

1    UT = Tasks           // unsigned tasks
2    TS = 0;             // timestamp
3    RUT_TS = {}          // sequence of possible to run unassigned Tasks
4    do
5     RUT_TS = getApplicableTasks (UT, TS)
6     if ( |RUT_TS| > 0 )
7       RUT_TS:= SORT( RUT_TS, SORT_CRITERIA )
8       RUT_TS:= getFirstNElements( RUT_TS, SIZE)
9       AssignedTasksMax = 0;
10      for all P permutation RUT_TS
11        numberOfAssignedTasks = 0
12        for each T_j task RUT_TS
13           Task = RUT.getFirstTask;
14           Tech = getShortDurAppTechn( Task, TS )
15           assigned = schedule.assign (Task, Tech, TS)
16           if (assigned)  numberOfAssignedTasks++
17        for end
18
19        if ( numberOfAssignedTasks > AssignedTasksMax )
20          AssignedTasksMax = numberOfAssignedTasks
21          BestTaskSequence = P
22
23        schedule.unassignTasks( RUT_TS )
24      end for
25
26      for each Task from BestTaskSequence
27        Tech = getShortDurnApplTech( Task, TS )
28        assigned = schedule.assign (Task, Tech, TS)
29        if (assigned) UT = UT / Tech
30      end for
31    end if
32    TS++;
33   while ( |UT| > 0)
```

schedule. As evaluation can be applied makespan or other schedule measure. In this paper model RCPSP some NEH modification must be implemented.

The basic version of NEH heuristic builds schedule partially to find optimal sequence of tasks adding next task to partial schedule, finally composing the whole schedule. In our approach NEH is considered rather as metaheuristics that proposes sequence of tasks that make schedule optimal (see Pseudocode 3). The other algorithm schedules task to build partial schedule – we decided to use the classical greedy algorithm. The best task sequence is marked as base task sequence (line 16), that is extended by next tasks probing all positions in the task sequence. Let's analyze the NEH working illustration. Having task A and task B, NEH executes *greedyAlgorithm* to find optimal tasks sequence (AB or BA). Let's assume that BA is optimal, to extend sequence BA adding new task C the *greedyAlgorithm* probes sequences: CBA, BCA and BAC and so on.

The basic NEH procedure is too time-consuming to apply in real world application. The next step of NEH implementation was to optimize its computational complexity. The most expensive operation is *GreedyAlgorithm* and this factor should be reduced. We observed that *GreedyAlgorithm* builds each time the whole schedule which is a huge extravagance. The next step was to use partially build schedule and reverse task sequence build strategy.

Pseudocode 3. NEH2 pseudocode

```
    procedure NEH2 (SORT_CRITERIA)

1   UT = Tasks          // unsigned tasks
2   BestTS = <>         // best sequence according to MAKESPAN
3   CurrentTS = <>      // current (candidate) tasks sequence
4   BaseTS = <>         // base task sequence
5
6   UT = SORT( UT, SORT_CRITERIA )
7   BestTS = CurrentTS = UT.getFirstTask()
8   for each Task from UT
9    CurrentTS = BaseTS
10   for each position i=|CurrentTS| insertion Task into CurrentTS
11              for each position CurrentTS to i
12                    rTask = CurrentTS.removeTask(i)
13                    ReTasks.addTask(rTask)
14             end for
15             value = GreedyAlgorithm(CurrentTS, ReTasks, Task)
16     if (value < bestValue or i==|CurrentTS|)
17       bestValue = value
18       BestTS = CurrentTS + Task + ReTasks
19     end if
20   BaseTS = BestTS
21  end for
```

For example, in basic NEH for three task (A, B and C, let assume that BA sequence is an optimal) the analyzed sequences are: CBA, BCA and BAC. In reverse order in NEH2 we build BA schedule as base, then BAC. In next sequence BCA from schedule is removed task A, then inserted C and A. In basic NEH to examine three task sequence 9 task is scheduled, in reduced version (NEH2, see Pseudocode 4) only 6 tasks is (re)scheduled.

The main modification of NEH2 heuristic is to remove from the initial sequence and reschedule only tasks that are next inserted (see line 11-14). To evaluate the partial schedule is build by *GreedyAlgorithm* (line 15) to examine the sequence of tasks.

The NEH2 computation complexity reduction makes possible practical application of heuristic in simpler cases. Such NEH2 heuristic has been examined. Results of test are presented in the next section.

## IV. Experiments and results

The t-RCPSP model is specialized to MP2 company requirements. All proposed methods in verification procedure need an empirical data. We analyzed real data and prepared dataset that is complete for the domain: various number of tasks, machines, devices and technologies. Such dataset allows us to do research and compare results of proposed methods.

All experiments are implemented in standard C/C++. Machine for test was equipped with Intel Core2 Duo 2.53 GHz, 4GB RAM and Windows7 OS. For each experiment, only one Core was used.

### A. *Experiments' set-up and dataset*

Prepared dataset MP2dataset[1] consists of seven various types of configurations – summary of dataset is presented in Table1. There are three types of simple settings (10_3x3_10, 50_10x20_40 and 75_10x20_40) that involve small number

Table I.
Summary of testing dataset MP2dataset

| | task s | machines | devices | technol ogies |
|---|---|---|---|---|
| **10_3x3_10** | 10 | 3 | 3 | 10 |
| **50_10x20_40** | 50 | 10 | 20 | 40 |
| **75_10x20_40** | 75 | 10 | 20 | 40 |
| **100_30x30_100** | 100 | 30 | 30 | 100 |
| **200_30x30_100** | 200 | 30 | 30 | 100 |
| **300_30x100_500** | 300 | 30 | 100 | 500 |
| **500_30x45_100** | 500 | 30 | 45 | 100 |
| Legend: tasks _ machines x devices _ technologies | | | | |

[1] MP2dataset is published in http://imopse.ii.pwr.edu.pl/

of tasks (respectively 10 or 50) and small number of devices, less than 20. Two configurations have medium difficulty (100_30x30_100 and 200_30x30_100) where number of tasks is larger (100 or 200) and there is increased number of possible technologies to 100. Additionally, the configuration 300_30x100_500 is difficult because of large number of tasks and extremely a lot of technologies (500) and devices (100). The last configuration consists of 500 tasks, which is the most difficult for methods testing.

To get dataset more general, for each configuration 100 instances were generated. The data generator constructs instances randomly according to the specific domain requirements and configuration. Analyzing the real data we assumed some additional dataset parameters: task deadline *ddt* in <10,50> defined in discrete timeslots, the maximal number of generated products by technology is 5. Each technology can produce no more that 2 types of products and use no more than 4 types of materials. Each product can be generated by 2 or more technologies. The longest technology duration not exceeds 10 timeslots.

## B. *Experiments results*

The main goal of provided experiments was to investigate how presented methods are effective in solving t-RCPSP. The method's results are described by makespan and averaged latency of all tasks in given schedule. The other comparative aspect was computational CPU time needed by methods to obtain results. All methods were investigated using MP2dataset and results were averaged to compare to others (see Table II). Research consists all examined methods DBH, NEH2 and PEC using one main parameter, sorting criteria: by task deadline ascending, descending and random. The PEC uses extra parameter: *size* of task list.

Experiments results presented in Table II give information that all developed heuristics are useful in solving t-RCPSP. In 4/7 cases the minimal makespan provided DBH, however NEH2 in such cases gives slightly worse results and in 3/5

TABLE II.
AVERAGED SCHEDULE MAKESPAN (AND STANDARD DEVIATION) FOR MP2DATASET

| Tasks_mach. | **10_3** | **50_10** | **75_10** | **100_30** | **200_30** | **300_30** | **500_30** |
|---|---|---|---|---|---|---|---|
| device_techn | **x3_10** | **x20_40** | **x20_40** | **x30_100** | **x30_100** | **x100_500** | **x45_100** |
| **DBH asc** | 185,53 | 189,43 | 252,65 | 190,21 | 336,88 | **214,72** | 899,0 |
| | 41,93 | 45,52 | 55,36 | 45,13 | 73,93 | **22,16** | 231,76 |
| **DBH dsc** | **184,9** | **183,9** | 252,5 | 190,4 | 333,4 | **214,7** | **880,0** |
| | **41,7** | **48,3** | 61,2 | 41,4 | 69,9 | **18,5** | **236,4** |
| **DBH rand** | 191,31 | 192,84 | 253,57 | 194,12 | 338,48 | 217,07 | 920,0 |
| | 42 | 48 | 58 | 45,9 | 72,95 | 20,04 | 250 |
| **PEC(3) asc** | 190,6 | 189,4 | **250,1** | 196,1 | 345,6 | 271,5 | 915,9 |
| | 43,0 | 43,9 | **53,5** | 48,1 | 68,0 | 18,9 | 251,5 |
| **PEC(3) dsc** | 189,6 | 185,0 | 254,8 | 192,2 | 342,1 | 273,3 | 906,4 |
| | 41,9 | 47,1 | 60,1 | 41,9 | 73,3 | 17,9 | 229,1 |
| **PEC(3) rand** | 193,0 | 189,44 | 256,87 | 196,33 | 339,9 | 257,63 | 927,51 |
| | 43 | 46 | 54 | 44 | 70 | 19,6 | 242 |
| **PEC(4) asc** | 187,28 | 187,65 | **250,56** | 192,43 | 342,08 | 259,6 | 926,0 |
| | 40 | 44 | **58** | 45 | 73 | 19,87 | 245 |
| **PEC(4) dsc** | 187,1 | 185,55 | 251,21 | 191,45 | 340,80 | 260,83 | 894,6 |
| | 43,9 | 49,14 | 59,04 | 39,21 | 70,05 | 17,74 | 232,42 |
| **PEC(5) asc** | 187,04 | 190,2 | 251,78 | 192,14 | 344,57 | 249,11 | 920,65 |
| | 39,04 | 41 | 59 | 49,9 | 71,93 | 19,29 | 242 |
| **PEC(5) dsc** | 190,02 | 185,78 | 254,4 | 191,0 | 339,09 | 245,78 | 896,9 |
| | 41,85 | 48,36 | 61 | 40,8 | 71,67 | 17,53 | 226 |
| **NEH2 asc** | 185,5 | 189,3 | 252,7 | **189,7** | 336,8 | *time limit exceeded* | *time limit exceeded* |
| | 41,9 | 45,7 | 55,2 | **46,1** | 73,0 | | |
| **NEH2 dsc** | 185,17 | **183,89** | 252,72 | **189,38** | **331,84** | *time limit exceeded* | *time limit exceeded* |
| | 41,29 | **48,33** | 61,12 | **41,28** | **69,3** | | |

TABLE III.
AVERAGED SCHEDULE LATENCY (AND STANDARD DEVIATION) FOR MP2DATASET

| Tasks_mach. device_techn | **10_3 x3_10** | **50_10 x20_40** | **75_10 x20_40** | **100_30 x30_100** | **200_30 x30_100** | **300_30 x100_500** | **500_30 x45_100** |
|---|---|---|---|---|---|---|---|
| **DBH asc** | 0,30 | 0,53 | 2,03 | 0,43 | **3,45** | 0,12 | **44,125** |
| | 0,52 | 0,89 | 2,32 | 0,79 | **2,57** | 0,15 | **19,70** |
| **DBH dsc** | 2,39 | 3,65 | 8,66 | 2,57 | 10,48 | 7,76 | 74,31 |
| | 1,60 | 3,19 | 4,37 | 2,05 | 4,06 | 2,00 | 23,53 |
| **DBH rand** | 1,6 | 2,2 | 5,83 | 1,62 | 7,63 | 4,70 | 63,32 |
| | 1,37 | 2,25 | 3,83 | 1,42 | 3,59 | 1,57 | 19,9 |
| **PEC(3) asc** | 0,29 | 0,42 | 2,05 | 0,52 | 3,88 | **0,10** | 49,64 |
| | 0,5 | 0,9 | 2,27 | 1,03 | 2,59 | **0,15** | 21,10 |
| **PEC(3) dsc** | 2,72 | 3,80 | 9,13 | 2,82 | 12,13 | 21,97 | 96,07 |
| | 1,59 | 3,26 | 4,56 | 2,07 | 4,2 | 3,45 | 20,85 |
| **PEC(3) rand** | 1,73 | 2,58 | 5,93 | 1,82 | 8,1 | 9,0 | 68,12 |
| | 1,4 | 2,82 | 3,47 | 1,62 | 3,5 | 2,2 | 19 |
| **PEC(4) asc** | **0,26** | 0,47 | 2,02 | 0,5 | 3,72 | 0,13 | 48,33 |
| | **0,44** | 0,92 | 2,75 | 1,0 | 2,72 | 0,19 | 20,65 |
| **PEC(4) dsc** | 2,53 | 3,94 | 8,94 | 2,64 | 11,64 | 18,65 | 90,27 |
| | 1,63 | 4,53 | 4,53 | 2,00 | 4,44 | 3,2 | 21,81 |
| **PEC(5) asc** | 0,28 | **0,40** | **2,00** | 0,47 | 3,52 | 0,14 | 47,34 |
| | 0,53 | **0,74** | **2,57** | 0,9 | 2,6 | 0,17 | 19,57 |
| **PEC(5) dsc** | 2,6 | 3,85 | 9,05 | 2,63 | 11,13 | 15,34 | 87,10 |
| | 1,6 | 3,22 | 4,43 | 2,03 | 4,22 | 2,94 | 21,0 |
| **NEH2 asc** | 0,30 | 0,52 | 2,05 | **0,42** | 3,47 | *time limit exceeded* | *time limit exceeded* |
| | 0,52 | 0,89 | 2,33 | **0,76** | 2,6 | | |
| **NEH2 dsc** | 2,39 | 3,65 | 8,66 | 2,55 | 10,5 | *time limit exceeded* | *time limit exceeded* |
| | 1,59 | 3,19 | 4,35 | 2,04 | 4,11 | | |

TABLE IV.
AVERAGED COMPUTATIONAL TIME [S] FOR MP2DATASET

| Tasks_mach. device_techn | **10_3 x3_10** | **50_10 x20_40** | **75_10 x20_40** | **100_30 x30_100** | **200_30 x30_100** | **300_30 x100_500** | **500_30 x45_100** |
|---|---|---|---|---|---|---|---|
| **DBH** | 0,14 | 0,05 | 0,12 | 0,13 | 0,57 | 4,8 | 7,13 |
| **PEC(3)** | 0,11 | 0,05 | 0,12 | 0,08 | 0,39 | 2,58 | 4,11 |
| **PEC(4)** | 0,10 | 0,05 | 0,12 | 0,1 | 0,41 | 2,95 | 4,71 |
| **PEC(5)** | 0,24 | 0,11 | 0,27 | 0,20 | 0,87 | 6,2 | 9,3 |
| **NEH2** | 5,6 | 1,2 | 3,9 | 5,6 | 49,66 | *time limit exceeded* | *time limit exceeded* |

cases returns solutions that compete with others. The NEH2 in more complicated cases (300 and 500 tasks) execution time exceeded 1 minutes CPU time and results are not taken into consideration. PEC only in one case returns the best solution (75 tasks, PEC(3) asc). Increasing the PEC *size* parameter value in most cases reduces makespan, e.g. comparing results of PEC(3) and PEC(4) using descending task deadline order. To summary results of all methods for all instances: DBH dsc needs average 319,97 timeslots to execute all 700 instances, DBH asc needs 324,06 timeslots and PEC(5) dsc has the third place: 328,9 timeslots. The longest averaged makespan schedule (equals to 337 timeslots) achieved PEC(3) heuristic with ascending task order. Analysis of the results presented into Table II can draw the conclusion that descending sorting criteria of tasks gives better results in the minimization of schedule makespan. Random tasks order makes solution the worst in all investigated cases. All methods are deterministic, the averaged results are computed on 100 instances of each configuration. In case of random tasks order, results for each instances are repeated 10 times and then averaged.

Results presented in Table III describe how generated schedules are late using as measure the average latency of all tasks in the schedule. The gained results proved our intuition that the best results give ascending sorting criteria of tasks – task with the shorter deadline is taken into consideration earlier. All methods showed that are effective, but it is rather impossible to point the best one. In 2/7 cases DBH gives the best solution, PEC in 4/7 cases (using *size* parameter equals to 3, 4 or 5). Results gained by NEH2 are not qualitative. Moreover, NEH2 in one case returns the best solution (100 tasks configuration). Comparing the averaged latency for 700 instances (whole MP2dataset) the best method gives 7,28 latency (DBH asc), the second one 7,73 (PEC(5) asc) and third one 7,91 (PEC(4) asc). The worst averaged latency achieved PEC(3) dsc: 21,23.

Comparing methods' working time (see Table IV) it is worth mentioning that methods are fast and effective. The provided MP2dataset of 700 instances gives an opportunity to compare methods results and recommend them to real-world applications. Increasing size of the problem, methods are practical as computational time not exceeds 10 seconds. Such short computational time makes possible to run several methods to get set of schedules and give a human operator a real choice.

The computation complexity of presented heuristics is $O(k^2)$ for DBH and $O(size!k^2)$ for PEC. The NEH2 complexity is larger because core of NEH2 is $O(k^2)$ but in each step uses *greedyAlgorithm* that is $O(k)$, what gives finally NEH2 $O(k^3)$. The result is that the computational time of NEH2 increases so dramatically that needed time is unacceptable in construction process of schedules that consist of more that 200 tasks. In such cases other methods are more effective and less demanding for CPU working time.

## V. Summary

Standard RCPSP, in presented work, was extended by technologies to solve practical problem in Plastic and Rubber Processing Industry – we defined t-RCPSP. Such model makes possible, in simply and intuitive way, a formalization of real-world problem. Each technology links non-renewable and renewable resources, uses various types of ingredients in production. Technologies that produce the same products may use resources in different way, more or less effectively. It can case be other production time consumption, too. The technology-driven RCPSP model gives a lot of possibilities to build effective heuristics. We proposed three of them: NEH2 adaptation, locally optimal resource usage PEC and duration driven heuristic DBH. Analyzing a real production schedule instances we implemented a data generator to get the MP2dataset (published in Internet) that includes 700 instances in 7 basic problem configurations to empirically prove efficiency of proposed methods. Results of experiments showed that proposed methods can be used as effective tool for scheduling in production company. Moreover, the next step was done: all presented methods were developed as automated scheduling module in MP2 company computer system.

### *A. Further research*

As the practical aspect of further research is the definition of more domain-based measures of final schedule and used technologies. In presented paper, only the makespan and latency are analyzed as the measure. However, in practice such technology can be more energy consuming, may need more labor (including human work) or can be less effective as a scrap measure is considered. The existence of several measures of schedule leads to the situation when multiobjective optimisation should be considered.

Investigating the comparison of presented methods we can see some possible directions of further work. The most promising is using metaheuristics (such as Evolutionary Algorithms or Tabu Search) to build schedule near optimal in cost/time criterion. Metaheuristics usage needs more computational time than simple heuristic – but our experience (e.g. Error: Reference source not foundError: Reference source not found) shows that results are (sub)optimal. Additionally, metaheuristics are using evaluation function (as the superposition of several schedule measures) can provide schedule dedicated to given user. Especially Tabu Search [12][15][16][20] application to RCPSP is very strong trend in literature.

## Acknowledgment

The research has been partially supported as a part of the Support Programme of the Partnership between Higher Education and Science and Business Activity Sector finances by City of Wroclaw.

## References

[1] Blazewicz J., Lenstra J.K., Rinnooy Kan A.H.G.; Scheduling subject to resource constraints: Classification and complexity, Discrete Applied Mathematics (5), pp. 11-24, 1983.

[2] Bouleimen K., Lecocq H.; A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version, Eur. J of Operational Research (149), pp. 268-281, 2003.

[3] Brucker P., Drexl A., Mohring R., Neumann K., Pesch E.; Resource–constrained project scheduling: Notation, classification, models, and methods, European Journal of Oper. Research (112), pp. 3–41, 1998.

[4] Dorigo M.; Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem, IEEE Transactions of Evolutionary Computation (1/1), pp. 53-66, 1997.

[5] Hartmann S.; A competitive genetic algorithm for resource–constrained project scheduling, Naval Research Logistics (45), pp. 733–750, 1998.

[6] Hartmann S., Briskorn D., A survey of variants and extensions of the resource-constrained project scheduling problem, European Journal of Operational Research 207(2010), pp.1-14. 2010.

[7] Kolisch R., Hartmann S., Experimental evaluation of state-of-the-art heuristics for the resource- constrained project scheduling problem, European Journal of Oper. Research (127), pp. 394–407, 2000.

[8] Kolisch R., Hartmann S., Experimental investigation of heuristics for resource-constrained project scheduling: An update, Euro. Journal of Oper. Research (174), pp. 23-37, 2006.

[9] Liang Y., Chen A., Kao W., Chyu C.; An Ant Colony approach to Resource–Constrained Project Scheduling Problems, Proc of the 5th Asia Pacific Indust. Eng. and Manag Systems Conf 2004, pp. 31.5.1-31.5.10, 2004.

[10] Luo S., Wang C., Wang J.; Ant Colony Optimization for Resource-Constrained Project Scheduling with Generalized Precedence Relations, Proc of the 15th IEEE International Conference on Tools with A(ICTAI03), pp. 284–289, 2003.

[11] Merkle D., Mittendorf M., Schmeck H.; Ant Colony Optimization for Resource–Constrained Project Scheduling, IEEE Transactions on Evolutionary Computation (6/4), pp. 333–346, 2002.

[12] Myszkowski P.B., Skowroński M. E., Myszkowski P. B., Kwiatek P., Adamski M., Tabu Search approach for Multi-Skill Resource-Constrained Project Scheduling Problem, Annals of Computer Science and Information Systems Volume 1, Proc. of the 2013 FeDCSIS Confeences, pp. 153-158, 2013.

[13] Myszkowski P.B., Skowronski M.E., Olech Ł.P. and Oślizło K., Hybrid ant colony optimization in solving multi-skill resource-constrained project scheduling problem, Soft Computing Journal, Sep 2014.

[14] Nawaz, M., Enscore, J., Ham, I.: A Heuristic Algorithm for the M-machine, N-task Flow-shop Sequencing Problem. Omega-Int. J. Manage. S. 11(1), 91–95 (1983)

[15] Pan H.I., Hsaio P.W., Chen K.Y.; A study of project scheduling optimization using Tabu Search algorithm, Engineering Applications of Artificial Intelligence (21), pp. 1101-1112, 2008.

[16] Pan N.H., Lee M.L., Chen K.Y.; Improved Tabu Search Algorithm Application in RCPSP, Proceedings of the International MultiConference of Engineers and Computer Scientists (Vol I), 2009.

[17] Santos M., Tereso A. P.; On the multi-mode, multi-skill resource constrained project scheduling problem - computational results, Soft Computing in Industrial Applications, Advances in Intelligent and Soft Computing (96), pp. 239–248, 2011.

[18] Skowroński M. E., Myszkowski P. B., Specialized genetic operators for Multi-Skill Resource-Constrained Project Scheduling Problem, 19th Inter. Conference on Soft Computing Mendel 2013, pp. 57-62, 2013.

[19] Skowroński M. E., Myszkowski P. B., Podlodowski L., Novel heuristic solutions for Multi-Skill Resource- Constrained Project Scheduling Problem, Annals of Computer Science and Information Systems Volume 1, Proc. of the 2013 Federated Conference on Computer Science and Information Systems, pp. 159-166, 2013.

[20] Thomas P. R., Salhi S.; A Tabu Search Approach for the Resource Constrained Project Scheduling Problem, Journal of Heuristics (4), pp. 123-139, 1998.

[21] Valls V., Ballestin F., Quintanilla S.; A hybrid genetic algorithm for the resource–constrained project scheduling problem, European Journal of Operational Research (185), pp. 495-508, 2008.

[22] Zhang H., Xu H., Peng W., A Genetic Algorithm for Solving RCPSP, 2008 International Symposium on Computer Science and Computational Technology, pp. 246–249, 2008.

[23] Zhang H., Li H., Tam C.; Particle swarm optimization for resource–constrained project scheduling, In- ter. Jour. of Project Management (24), pp. 83-92, 2006.

[24] Zhang K., Zhao G., Jiang J.; Particle Swarm Optimization Method for Resource-Constrained Project Scheduling Problem, The Ninth International Conference on Electronic Measurement & Instruments ICEMI2009, pp. 792–796, 2009.

[25] Ziarati K., Akbari R., Zeighami V.; On the performance of bee algorithms for resource–constrained project scheduling problem, Applied Soft Computing (11), pp. 3720–3733, 2011