Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 50-1 (2017) 7970–7974

Long-term production planning problem: scheduling, makespan estimation and bottleneck analysis.

Dmitry I. Arkhipov * **Olga Battaïa** ** **Alexander A. Lazarev** ***

* *Department of Complex Systems Engineering, ISAE-SUPAERO, Universite de Toulouse, 10 avenue Edouard Belin - BP 54032 - 31055 Toulouse Cedex 4 France; V.A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences, Moscow, Russian Federation. (e-mail: miptrafter@gmail.com)*

** *Department of Complex Systems Engineering, ISAE-SUPAERO, Universite de Toulouse, 10 avenue Edouard Belin - BP 54032 - 31055 Toulouse Cedex 4 France. (e-mail: olga.battaia@isae.fr)*

*** *V.A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences, Moscow, Russian Federation; Lomonosov Moscow State University, Moscow, Russian Federation; Moscow Institute of Physics and Technology, Dolgoprudny, Russian Federation; International Laboratory of Decision Choice and Analysis, National Research University Higher School of Economics, Moscow, Russian Federation. (e-mail: jobmath@mail.ru)*

Abstract: In this paper, a long-term production planning problem is considered with the objective criteria C_{max} and T_{max} and under resource capacity and precedence constraints. The case study presented is characterized by four years planning horizon, 3552 operations, 51 workers and 57 units of equipment. The solution method elaborated in this study is a heuristic algorithm. Its performances are evaluated in numerical experiments. New procedures for makespan and resource load estimation are developed in order to identify bottleneck resources. The makespan estimation algorithm is tested on the well-known PSPLIB benchmark library where the best known lower bounds are improved for 5 instances. This procedure can also be used for the estimation of the gap from optimal value of the makespan time provided by the heuristic algorithm.

Keywords: Scheduling, resource-constrained project scheduling, timetabling, heuristics, makespan, bottleneck analysis

1. INTRODUCTION

We consider the problem of long-term planning for an engine assembly company. This problem can be considered as a special case of Resource Constrained Project Scheduling Problem (RCPSP) with a huge number of operations and two types of renewable resources. However, the considered problem has also some particularities which make it even harder than the classical formulation of RCPSP. First difference lies in the fact that the resource consumption rate for an operation depends on the resource type. Another difference is due to the need to create timetables for resources. Because of long-term planning horizon, the problem instances can be very large with a huge number of operations, workers and pieces of equipment. This study was motivated by the real-data problem with 3552 operations, 51 workers, 57 units of equipment and four years planning horizon. Taking into account such voluminous data that can be even more significant for some instances, heuristic approach was chosen to guarantee an optimization result in reasonable solution time.

RCPSP is a well-studied classical combinatorial optimisation problem. Garey & Johnson (1975) proved that the decision variant of the RCPSP is NP-complete in the strong sense even without precedence constraints and only one resource, by reduction to the 3-partition problem. A comprehensive survey on project scheduling problems formulations and solution methods was presented by Kolish & Padman (1997). Most of makespan lower bound estimation approaches are listed in the surveys written by Neron et.

⋆ This research is supported by the Russian Science Foundation (grant 17-19-01665).

10.1016/j.ifacol.2017.08.991

al. (2006) and Knust (2015). To compare algorithms performance and obtained lower bounds, benchmark library PSPLIB was created by Kolisch & Sprecher (1997). For the most of PSPLIB instances the best known makespan lower bounds were provided by the approaches presented by Brucker & Knust (2000), Schutt et. al. (2013) and Berthold et. al. (2010). Experimental evaluations of existed algorithms were presented in Hartmann & Kolisch (2000) and Kolisch & Hartmann (1998).

Since the problem considered in this paper differs from the classical formulation and in particular by its large scale, a new efficient heuristic approach is developed. This paper is organised as follows. The problem formulation and heuristic algorithm are presented in Sections 2 and 3 respectively. Section 4 describes the approach to estimate makespan and identify bottleneck resources. The results of numerical experiments are presented in Section 5 and some concluding remarks are given in Section 6. In the Section the table of most used notations is presented.

2. PROBLEM FORMULATION

The following optimization problem is considered. There is a set of orders O. For each order $i \in O$ a due date D_i is given. All orders are presented at time $t = 0$. For each ordered product there is a set of operations N_i and an assembly scheme presented by direct tree graph $G_i(N_i, TI_i)$ of operations to be done. All vertices of the graph G_i have only one outgoing edge, except the *final* vertex, which has only incoming edges. We denote the union of such graphs for all orders of the set O by $G(N, I)$, i.e. N is the set of operations to be done to perform all ordered products. For each operation $j \in N$, the following parameters are defined:

- set of required worker's skills RW_j;
- set of necessary equipment RE_j;
- worker occupation time p_j^w;
- equipment consumption time p_j^e.

One of the difference in comparison with classical RCPSP formulation is that for some operations $j \in N$ the inequality $p_j^w > p_j^e$ holds (Fig. 1). The sets of workers W and equipment E are given. One worker can have only one skill. The objective is to assign workers and equipment to

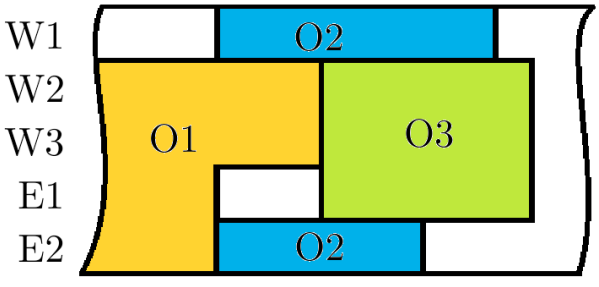

Fig. 1. Production of operations $O1$, $O2$, $O3$ by workers $W1$, $W2$, $W3$ using equipment $E1$ and $E2$.

operations and set up start time S_j and finish time F_j for each operation $j \in N$ subject to precedence relations and to minimize the bi-criteria of minimal tardiness and makespan objective functions

$$\min \max_{j \in N} F_j | \min \max_{i \in O} \{0, \max_{k \in N_i} \{F_k - D_i\}\}.$$

Heuristic approach is based on the idea of scheduling construction from right to left, minimizing the idleness of workers and equipment on each step, subject to critical paths of the graph G.

3. HEURISTIC APPROACH

3.1 Additional notations.

Let us introduce additional notations. For any operation $j \in N$ we define

- critical path $P_j = \max_{K \in seq_j} \sum_{l \in K} \max\{p_l^w, p_l^e\}$, where seq_j is a sequence of operations related to the path of the graph G incoming into vertex j and $j \in seq_j$;
- set of previous operations $prev_j$ is defined by graph G;
- one next operation $next_j$ is defined by graph G for all operations except final operations of orders of set O;
- state of operation s_j equals to:
  -1, if operation $next_j$ is not processed yet,
  0, if $next_j$ is processed, but s_j is still not processed,
  1, if s_j is already done.
  The initial state of operations equals to 0 for any final operation of the ordered product (if $next_j = \emptyset$) and equals -1 for others;
- operation ready time r_j equals D_i if j is a final operation of order O_i and equals S_{next_j} if $s_j \neq -1$ for other vertices. If $s_j = -1$, then r_j is not defined.

At each step of the algorithm, a set Q consists of operations with state $s_j = 0$.

For any equipment $k \in E$ or worker $l \in W$, we define a set of time intervals where it cannot process operations. We denote it as TI_k^e and TI_l^w respectively. Initially all sets consist of only one interval $[\max_{i \in O} D_i, +\infty)$.

3.2 Algorithm description

Now let us present an approach to find suboptimal solution of the stated problem. It consists of two parts: inner cycle Algorithm ICA and main Algorithm MCA. ICA constructs the schedule for a set of heuristic parameters x_1, x_2, x_3, and the main Algorithm changes x_1, x_2, x_3 and chooses the solution with the best objective function value.

Inner cycle Algorithm description. Firstly we set states s_j of all operations $j \in N$: $s_j := 0$ if $next_j = \emptyset$ and $s_j := -1$ otherwise. Then we start an iteration cycle. At each iteration the set of candidates Q consists of all operations $j \in N$ with $s_j = 0$. Then operation $j \in Q$ with the highest value of criterion $CR(j)$ is chosen to be processed. Moments of time S_j and F_j, arrays of workers W_j and equipment E_j for processing operation j are found during the process of calculating $CR(j)$. After that algorithm changes the state of operation to $s_j := 1$ and for each previous operation $k \in prev_j$ set up state $s_k := 0$ and ready time $r_k := S_j$. For any worker $i \in W_j$ we add an interval $[S_j, S_j + p_j^w)$ to set TI_i^w. For any equipment $i \in E_j$ we add an interval $[S_j, S_j + p_j^e)$ to set TI_i^e. Then

we collect statistical data (S_j, F_j, W_j, E_j) and go to the next iteration. Algorithm terminates when all operations are executed, i.e. for all $j \in N$ equality $s_j = 1$ holds.

The most important part of ICA is the calculation of criterion $CR(j)$.

Criterion CR(j).

There is a lot of ways to formulate heuristic criterion to choose an operation and a set of workers to process it. Let's list some motivations to choose an operation j and a set of occupied workers W_j and equipment E_j at an inner cycle iteration.

1. Higher S_j provides later ready times for operations of set $prev_j$. This helps to satisfy due dates.
2. High cardinality of set $prev_j$ increases the number of candidate operations at the following iterations.
3. High critical path P_j allows to satisfy due dates.
4. Usage of highly-demanded (subject to resource capacity) resources with low current load can prevent from the violation of due dates.

We can formulate the function which takes into account all motivations M_1, M_2, M_3, M_4 as follows. For solving the considered problem the following criterion is suggested:

$$CR(j) = |prev_j|(x_1 RLC + x_2 CPC - x_3 RTC),$$

where $x_1,\ x_2,\ x_3$ are heuristic parameters and

- RLC – resource load component, depends on the sum of ratios of non-processed operations demanded amount and candidate operations demanded amount for all required resources

$$RLC = \sum_{i \in E_j} \frac{\sum\limits_{l \in N | s_l = \{-1,0\}} p_l^e}{\sum\limits_{l \in N | s_l = \{0\}} p_l^e} + \sum_{i \in W_j} \frac{\sum\limits_{l \in N | s_l = \{-1,0\}} p_l^w}{\sum\limits_{l \in N | s_l = \{0\}} p_l^w}.$$

- CPC – critical path component depends on the critical path and the remaining time to the moment $t = 0$

$$CPC = \frac{1}{r_j - P_j}.$$

- RTC – ready time component determined by the difference between the highest ready time and r_j

$$RTC = \max_{l \in Q}\{r_l\} - r_j.$$

It is important to note that this criterion is dynamic, i.e. it depends on the parameters of unprocessed operations, candidate operations, resource load and remaining time. Therefore, the value $CR(j)$ for the same $j \in N$ is different at each step of ICA.

When operation j is chosen the latest possible moments of time S_j and F_j are calculated.

Main Algorithm description

On each iteration main Algorithm changes heuristic parameters x_1, x_2, x_3 and constructs the schedule using ICA. Then it chooses a schedule with the best objective function value. Main Algorithm can use a lot of different ways to change the parameters $x_1, \ldots, x_3$ (linear, quadratic, exponential, mixed, etc.) to find a better solution.

If for any due date D_i of any order $i \in O$ feasible solution is not found, MCA increases D_i to $D_i' = D_i + const$ and repeats ICA. If feasible solution has been found, MCA decreases D_i and tries to find a solution for new due dates. This process can be terminated by the achievement of desired precision of maximum tardiness value or by the end of fixed amount of computational time. We can change due dates using different approaches such as logarithmic search or a search with the defined constant value.

4. MAKESPAN ESTIMATION AND BOTTLENECK ANALYSIS

In this section the Algorithm of resource load analysis is presented. We define an upper bound on the amount of resource $i \in R$ consumed up to a moment of time t as $U_i(t)$. An upper bound on the resource $i \in R$ consumed amount up to moment of time t subject to resource k is denoted by $U_{i|k}(t)$. Number of resource i units required for operation $j \in N$ is defined by NR_j^i. A set of operations $j \in N$ such as $NR_j^i > 0$ is defined by N^i. The Algorithm UBE provides the estimation of $U_i(t)$.

Given data. For each resource $i \in R$

- set of operations N^i;
- capacity c_i.

For each operation $j \in N^i$ the following parameters are given

- release time rt_j, if it is not given, set rt_j equals to the incoming critical path P_j value;
- processing time p_j^k for any resource $k \in R$ such as $NR_j^i > 0$.

We suppose that any operation $j \in N$ requires NR_j^k units of a resource k in interval $[S_j, S_j + p_j^k)$. For any operation $j \in N^i$, we define an available demanded amount $DA_j^i(t)$ equals to

- 0 if $t < rt_j$,
- $NR_j^k \cdot (t - rt_j + 1)$, if $rt_j \le t \le rt_j + p_j - 1$,
- $NR_j^k \cdot p_j$, if $t > rt_j + p_j - 1$.

for a moment of time t equals We also define an amount of a resource $i \in R$ used by operation $j \in N^i$ as $UA_j^i(t)$, which initially equals 0 for any $t \ge -1$.

UBE Algorithm description

Consider all pairs of resources $i, k \in R$ for the chosen resource i. For any pair i, k algorithm starts at the moment of time $t = 0$ and repeats the following steps for $t = 0, 1, \ldots, t^*$ in order to find $U_{i|k}(t^*)$.

1. Update available demanded amount values for all $j \in N^i$. If there is an operation which holds
$$DA_j^i(t) - UA_j^i(t-1) > 0,$$
go to the step 2, otherwise increase $t := t + 1$. If $t > t^*$ terminate the algorithm and return value $U_{i|k}(t^*) := \sum\limits_{j \in N^i} UA_j^i(t)$, otherwise come back to Step 1.
2. Use the highest possible amounts of resources i, k by operations which hold
$$DA_j^i(t) - UA_j^i(t) > 0.$$
The utilization of resources i and k by operations of the set N must satisfy capacity constraints. If there

are several possible ways to use the highest possible amounts of a resource, choose the one with respect to criterion 1.

$$\min \sum_{j \in N} \sqrt{(UA_j^i(t) - UA_j^i(t-1))^2 + (UA_j^k(t) - UA_j^k(t-1))^2}. \quad (1)$$

3. If there are some operations $j \in N^i$ whose processing is not completed yet, i.e. $UA_j^i(t) < p_j^i \cdot NR_j^i$, then increase $t := t + 1$. If $t > t^*$, return value $U_{i|l}(t^*) := \sum_{j \in N^i} UA_j^i(t)$, otherwise go to Step 1.

When all values $U_{i|k}(t^*)$ are found for resource i and all k, the lowest one is returned, i.e.

$$U_i(t^*) = \min_{k \in R, k \neq i} U_{i|k}(t^*),$$

and algorithm terminates. The following theorem implies the correctness of UBE algorithm.

Theorem 1. Let all operations $j \in N$ in schedule π start their execution at the moment of time S_j and use required amount NR_j^i of any resource $i \in R$. Then, the total amount of resource i used in the interval $[0, t+1)$ is not more than $U_i(t^*)$, i.e.

$$\sum_{j \in N} NR_j^i \cdot \max\{0, \min\{t - S_j, p_j^i\}\} \leq U_i(t^*).$$

Makespan lower bound estimation

The obtained result lead us to a new makespan lower bound estimation Algorithm MEA. It can be formulated as follows.

1. Set initial time $t := 0$ and makespan lower bound $MLB = 0$ and go to step 2.
2. For each resource $i \in R$ do the following procedures.
   a) Find $U_i(t)$ using Algorithm UBE. Calculate not used amount of i using the following formula
   $$NU_i(t) = \sum_{j \in N} (NR_j^i \cdot p_j^i) - U_i(t).$$
   b) Estimate the $MLB_i^{right}(NU_i(t))$ – lower bound on the time required to use amount $NU_i(t)$ by executing operations from right to left. It can be done by an algorithm similar to UBE applied for the set of operations N' with the same parameters but opposite edges directions of the precedence graph G. The only difference of such an algorithm is that it does not terminate at a fixed moment of time t but when amount $NU_i(t)$ of resource i is consumed. If $NU_i(t) = 0$, set $MLB_i^{right}(NU_i(t)) = \min_{i \in N_i'} r_i$ to take into account operations which do not require resource i.
   c) If the obtained makespan lower bound equals to $t + 1 + MLB_i^{right}(NU_i(t))$ i.e. is greater han MLB, set $MLB = t + 1 + MLB_i^{right}(NU_i(t))$.
3. If for any resource $NU_i(t) > 0$, set $t := t + 1$ and go to Step 2. Otherwise, return MLB and terminate the Algorithm.

Theorem 2. For any set of operations N, acyclic direct precedence relation graph G and set of resources R, value MLB found by the MEA is a lower bound on makespan of considered problem instance.

Bottleneck analysis

Theorems 1 and 2 provide an upper bound on resource used amount and makespan lower bound. Makespan lower bound allows us to analyse the efficiency of constructed solution and estimate the gap of the obtained values of the objective functions C_{max} and T_{max}. Resources with the highest utilization rate can be considered as a bottleneck for the considered problem.

5. NUMERICAL EXPERIMENTS.

Makespan estimation algorithm was implemented using C++ programming language and tested on the industrial case study using processor Intel(R) Core(TM) i5-4670 3.40GHz and 8 GB of RAM. The case study instance was characterized by 3552 operations, 51 workers and 57 units of equipment. Table 1 presents a comparison of the suggested approach, multi-agent approach and optimal result, which was obtained by constraint programming model developed by Ruslan Sadykov. Makespan estimation algorithm returned 577 days as makespan lower bound. This represents 79% of the makespan optimal value.

Table 1. Industrial case numerical experiments results.

approach	tardiness (days)	makespan (days)
heuristic	101	791
multi-agent	102	792
constraint programming	40	730

Makespan estimation algorithm was also tested on a well-known PSPLIB benchmark library and for 5 instances it outperforms the existed approaches. The results are presented in Tab. 2.

Table 2. *tasks* – number of tasks, *instances* – number of tested instances, *NW BKLB* – percentage of instances, where the obtained lower bound is not worse than the best one, *MAX deviation* and *AVG deviation* – maximal and average deviation values respectively, where deviation = (best known lower bound – obtained lower bound)/(best known lower bound), *bounds improved* – number of improved bounds.

tasks	instances	NW BKLB %	MAX deviation %	AVG deviation %	improved
30	445	66,5	31,5	3,7	0
60	450	71,4	22,7	2,4	0
90	445	75,3	16,7	1,2	0
120	600	54,7	15,3	1,7	5

6. CONCLUSION

In this paper, a heuristic approach to solve a long-term production planning problem was presented. It was tested on the data provided by engines' assembly company. Algorithms to estimate a makespan lower bound and upper bounds for the resource utilization rates were also developed. By applying these algorithms, the lower bounds were improved for 5 instances from PSPLIB benchmark

library. Due to its pseudo-polynomial complexity, the algorithm can be applied to large-scale problem instances.

Future research will be focused on the improvement of the makespan estimation algorithm. It will be extended to more complex formulations of RCPSP, for example by considering non-renewable resources with time-dependent capacities and sets of "time windows" for each operation $j \in N$.

7. USED NOTATIONS

In Table 3 we summarize the notations used in the paper.

Table 3. Notations.

O	set of orders
D_i	due date of an order $i \in O$
N_i	set of operations to complete order $i \in O$
G_i	precedence relation graph for set N_i
E	set of equipment
W	set of workers
RE_j	set of equipment required to process operation j
RW_j	set of workers required to process operation j
p_j^e	equipment occupation time by the task j
p_j^w	worker occupation time by the task j
rt_j	release time of an operation j
P_j	length of the critical path starting from vertex j
S_j	start time of operation j
F_j	end time of operation j
s_j	state of operation j
$prev_j$	set of operations i such as an edge $i \to j$ exists in graph G
$next_j$	operation i such as an edge $j \to i$ exists in graph G
Q	set of "candidate" tasks defined at each iteration of ICA algorithm
NR_j^i	amount of resource $i \in E \cup W$ required to process operation j
$DA_j^i(t)$	highest possible amount of resource i demanded by operation j
$UA_j^i(t)$	amount of resource i used by operation j during time horizon $[0, t)$ in the algorithm UBE
$U_{i\|l}(t)$	upper bound of the utilization of resource i under condition of the utilization of resource l during time horizon $[0, t)$ obtained by the algorithm UBE
$U_i(t)$	upper bound on the amount of resource i which can be used by the operations in the interval $[0, t)$
$NU_i(t)$	lower bound on the amount of resource i which has to be used by the operations after time t

REFERENCES

M.R. Garey, D.S. Johnson Complexity results for multiprocessor scheduling under resource constraints. *SIAM Journal on Computing*. 4 (4), pp. 397411. doi:10.1137/0204035.

Kolish R., Padman R. An Integrated Survey of Project Scheduling. 1997.

Neron E., Artigues C., Baptiste P., Carlier J., Damay J., Demassey S., Laborie P. Lower bounds for Resource Constrained Project Scheduling Problem In: *Jozefowska J., Weglarz J.: Perspectives in Modern Project Scheduling*, Springer, 2006, pp. 167–204.

Knust S. Lower Bounds on the Minimum Project Duration. In: *Schwindt C., Zimmermann J.: Handbook on Project Management and Scheduling*, vol. 1, Springer, 2015, pp. 356.

Kolisch R., Sprecher A. PSPLIB – a project scheduling problem library. *Eur J Oper Res*, Vol. 96(1), 1997, pp.205–216, http://www.om-db.wi.tum.de/psplib/.

Brucker P., Knust S. A linear programming and constraint propagation-based lower bound for the RCPSP *European Journal of Operational Research* Vol. 127 (2), 2000, pp. 355-362.

Schutt A., Feydy T., Stuckey P., Wallace M. Solving RCPSP/max by lazy clause generation. *Journal of Scheduling*, Vol. 16(3), June 2013, pp. 273-289.

T. Berthold, S. Heinz, M.E. Lubbecke, R.H. Mohring and J. Schulz. A Constraint Integer Programming Approach for Resource-Constrained Project Scheduling. Proceedings of CPAIOR 2010.

Hartmann S., Kolisch R. Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 2000, Vol. 127, pp. 394-407.

Kolisch R., Hartmann S. Heuristic Algorithms for Solving the Resource-Constrained Project Scheduling Problem: Classification and Computational Analysis. *Manuskripte aus den Instituten fur Betriebswirtschaftslehre*, 1998, No. 469, Kiel, Germany.