Minimizing the number of workers for one cycle of a paced production line

Alexander Dolgui [*] **Sergey Kovalev** [**] **Mikhail Y. Kovalyov** [***] **Sergey Malyutin** [*] **Ameur Soukhal** [****]

[*] *Ecole des Mines de Saint-Etienne, FAYOL-EMSE, CNRS UMR 6158 LIMOS, 158, cours Fauriel, 42023 Saint Etienne Cedex 2, France, E-mail: {dolgui,sergey.malyutin}@emse.fr*
[**] *INSEEC Business School ECE Lyon, 19 place Tolozan, 69001 Lyon, France, E-mail: smkovalev@yahoo.com*
[***] *United Institute of Informatics Problems, National Academy of Sciences of Belarus, Minsk, Belarus, E-mail: kovalyov_my@newman.bas-net.by*
[****] *Laboratoire d'Informatique, Ecole Polytechnique de l'Université de Tours,Tours, France, E-mail: ameur.soukhal@univ-tours.fr*

Abstract: We consider a paced line that produces various products and consists of several assembly stations. Each product or semi-product requires a specific set of operations to be performed by identical workers. The assignment of operations to the stations and precedence relations between the operations are known. Operations assigned to different stations are performed simultaneously and those assigned to the same station are performed sequentially. No worker can perform more than one operation at a time. The processing time of an operation depends on the number of identical workers performing this operation. If a worker is assigned to an operation, he is busy with this task from its start till completion. The problem is to find a schedule, which specifies operation start times and assignment of workers to the operations, such that the line cycle time constraint and the number of workers box constraints for each operation are satisfied. We prove that the problem is NP-hard in the strong sense, suggest conventional and randomized heuristics, describe a reduction to a series of feasibility problems, show relation of the feasibility problem to multi-mode project scheduling and multiprocessor scheduling, and introduce a MILP model for the feasibility problem.

Keywords: algorithms, heuristics, production systems, integer programming, scheduling, optimization

1. INTRODUCTION

Consider a line that manufactures various products and consists of several (work)stations connected by transporting devices. The line is *paced*, which means that each station switches from processing a current (semi)product to the next at the same time and with the same time step. The time interval that spans between two successive switches is called *cycle* and its length is called *cycle time*. Motivated by a real life production of car engines, we study a workforce assignment problem for one cycle of the paced line. Without loss of generality, we assume that the cycle starts at time zero.

In a given cycle, a given set of operations is performed on each station. Operations of different stations are performed in parallel. If operation starts, it continues with no preemption until its completion. The technological process on each station is characterized by the given *precedence relations* between the operations. If operation i precedes operation j, then j cannot start earlier than the completion time of i. If there is no precedence relation between operations i and j, then these operations are called *independent* and they can be performed in parallel. For example, operations of different stations are independent. Operations that have no predecessor can start at time zero. The precedence relations are represented by a *directed acyclic graph* $G = (N, U)$, where N is the set of *vertices* (operations) and $U \subset N \times N$ is the set of *arcs* (ordered pairs of operations) (i, j) such that $(i, j) \in U$ if and only if operation i precedes operation j. Let $|N| = n$ and $|U| = u$.

Operations are performed by *identical* workers of a set $R_{\max} = \{1, \ldots, r_{\max}\}$. Each operation j is associated with a *processing time* $p_j(r)$ which is an integer positive non-increasing function of the number of workers r assigned to it. If a worker is assigned to an operation, he is fully occupied by this operation from its start till completion. No worker can perform any two operations at a time. In our practical case, the time of worker's movement between the stations is negligibly small comparing with the time of any operation. Therefore, we assume that any worker can move from one operation to another at zero time.

The decision to be made is a *schedule*, which specifies the start times of operations and an assignment of workers to operations over time. Given a schedule, we denote by r_j, S_j and C_j the number of workers assigned to operation

j, its *start* and *completion time*, respectively. We have $C_j = S_j + p_j(r_j)$, $j = 1, \dots, n$. The schedule *makespan* is defined as $C_{\max} = \max_{j \in N}\{C_j\}$. This value can also be viewed as the line cycle time.

A *feasible* schedule must satisfy the following constraints:

- $a_j \le r_j \le b_j$, where a_j and b_j are given positive integer numbers, $j = 1, \dots, n$ (box constraints), and
- $C_{\max} \le d$, $j = 1, \dots, n$, where d is a given upper bound on the line cycle time.

The problem is to find a feasible schedule which minimizes the maximum number of workers used simultaneously,

$$W_{\max} = \max_{0 \le t \le d} \Big\{ \sum_{j \in N(t)} r_j \Big\},$$

where $N(t)$ is the set of operations performed at time t. We denote this problem as $MinNumber$. Let $W^*_{\max}$ denote the minimum $W_{\max}$ value. Assume without loss of generality that $r_{\max} \le \sum_{i \in N} b_i$.

For an example, consider a problem with two stations and nine operations. Any number of workers can perform any operation but operations 2, 7 and 8, which should be performed by one worker. Precedence relations are given by the graph in Figure 1.

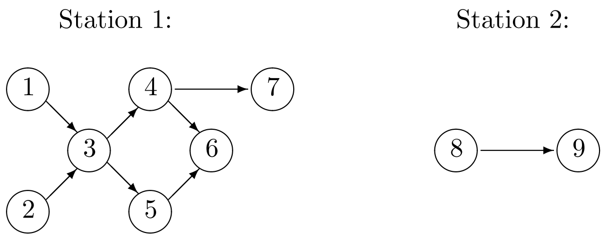

Fig. 1. Precedence graph

A diagram of a feasible schedule is given in Figure 2.

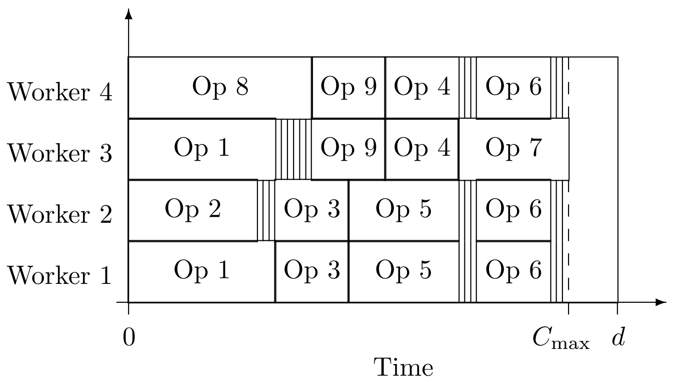

Fig. 2. A feasible schedule. Dashed rectangles represent idle times of workers.

Problems of workforce assignment to operations of a production line are intensively investigated lately. For example, see Blum and Mirales (2011), Karabak et al. (2011), Araujo et al. (2012), Moreira and Costa (2013), Mutlu et al. (2013), to name a few. To the best of our knowledge, a combination of workforce dependent operation processing times, box constraints on the number of workers and operation precedence constraints, which is specific for the problem $MinNumber$, has never been studied before.

2. COMPUTATIONAL COMPLEXITY

In this section, we first show that the problem $MinNumber$ is NP-hard in the strong sense, and then describe heuristic algorithms for it.

Note that the decision version of the problem $MinNumber$, in which there is one station, $a_j = b_j = 1$ for all operations, there are no precedence constraints and the upper bound on the number of workers is m, is equivalent to the decision version of the classic scheduling problem $Pm||C_{\max}$, which is NP-complete in the strong sense for variable m, see Garey and Johnson (1979). If, additionally, $m = 2$ and the precedence graph is a collection of chains, then it is equivalent to the decision version of the classic scheduling problem $P2|chains|C_{\max}$, which is NP-complete in the strong sense as well, see Du et al. (1991).

We will prove that $MinNumber$ is NP-hard in the strong sense even if there are no precedence constraints, values a_j and b_j are not unit, and operation processing times are inversely proportional to the number of assigned workers, as it is in the real-life case which motivates our studies.

Theorem 1. The problem $MinNumber$ is NP-hard in the strong sense even if there are no precedence constraints and $p_j(r) = p_j/r$, $j \in N$.

We will use a reduction from the NP-complete problem 3-Partition, see Garey and Johnson (1979).

3-Partition: Given $3k + 1$ positive integer numbers $h_1, \dots, h_{3k}$ and H satisfying $\sum_{j=1}^{3k} h_j = kH$, is there a partition of the set $\{1, \dots, 3k\}$ into subsets $X_1, \dots, X_k$ such that $\sum_{j \in X_t} h_j = H$ for $t = 1, \dots, k$? Assume without loss of generality that $h_j \ge k+1$, $j = 1, \dots, 3k$. If it is not the case, all numbers $h_1, \dots, h_{3k}$ and H can be multiplied by $k + 1$ with no change of the problem complexity.

Given an instance of 3-Partition, we construct an instance of $MinNumber$, in which there are $n = 3k$ operations, no precedence constraints, the number of stations plays no role, $p_j(r) = h_j/r$, $a_j = h_j - 1$, $b_j = h_j$, $j = 1, \dots, n$, and $d = k$. Thus, the processing time of operation j can take one of the two values: 1 or $1 + \frac{1}{h_j - 1}$, where $1 \le 1 + \frac{1}{h_j - 1} \le 1 + \frac{1}{k}$, $j = 1, \dots, n$. We will show that there exists a feasible solution for this instance with value $W^*_{\max} \le H$ if and only if the instance of 3-Partition has a solution.

Assume that there exists a feasible solution of the problem $MinNumber$ with r_j workers assigned to operation j, $j = 1, \dots, n$, and the maximum number of used workers $W^*_{\max} \le H$. For this solution, let us represent an assignment of a worker i to operation j as a *small rectangle* of height 1 and length $p_j(r_j)$ located in line i of a *large rectangle* of height H and length k. Then the solution can be viewed as a collection of small rectangles inscribed into the large rectangle so that no two small rectangles have a common point other than on their border. Figure 2 can be used for an illustration.

Observe that every operation j contributes h_j to the total square of the large rectangle irrespectively of the assignment of workers, because $rp_j(r) = h_j$ for any r. Since $\sum_{j=1}^{n} h_j = kH$, the union of the small non-intersecting rectangles must give the large rectangle.

We will now show that every operation j is assigned h_j workers. Assume the contrary that at least one operation q is assigned $h_q - 1$ workers. Consider one of these workers. Let it be used to fulfill e operations, and let I be a subset of these operations, each operation j of which is assigned $h_j - 1$ workers. We have $e \le k-1$, because if $e = k$. Then, since the processing time of every operation is at least one time unit and the processing time of at least one operation is greater than one time unit, the total occupation time of this worker is greater than k, that is, the cycle time constraint is violated. The total occupation time of this worker is equal to

$$e + \sum_{j \in I} \frac{1}{h_j - 1} \le k-1+\frac{|I|}{k} \le k-1+\frac{e}{k} \le k-1+\frac{k-1}{k} < k.$$

This strict inequality implies that there is a space in the large rectangle not occupied by any small rectangle, which is a contradiction.

We have shown that every operation j must be assigned h_j workers. Hence, the processing time of any operation is equal to 1. Denote the set of operations performed in parallel in the time interval $[t-1, t]$ as X_t. Since the union of small rectangles gives the large rectangle, equality $\sum_{j \in X_t} h_j = H$ holds for $t = 1, \ldots, k$, as required for the proof of the part "only if".

Part "if" is proved easily: if $X_1, \ldots, X_k$ is a solution of the problem 3-Partition, then assign h_j workers to operation j, $j = 1, \ldots, n$, and perform operations of the set X_t in parallel in the time interval $[t-1, t]$, $t = 1, \ldots, k$. □

Since the problem $MinNumber$ is NP-hard in the strong sense, a heuristic solution algorithm can be employed to solve it. If it finds a feasible solution, then the value of this solution can be used as an upper bound on the optimal number of workers.

3. HEURISTICS

We suggest two constructive heuristics for the problem $MinNumber$, one conventional and one randomized. Both heuristics construct a solution, in which consecutively numbered workers are assigned to the same operation. In the beginning, both heuristics determine a total number workers, W, to be used, determine numbers of workers for each operation: $r_j = a_j$, $j \in N$, and construct a schedule by assigning operations to W workers over time in an order, which is topologically feasible with respect to the precedence graph G. As there can be many topologically feasible orders, the conventional heuristic, called *TopLong*, selects a longest operation first, and the randomized heuristic, called *TopRandom*, selects an operation to be assigned first at random.

Heuristic $TopLong$ (conventional):

Step 1 (Initial values of W and r_j) Determine an initial total number of workers, W. Clearly, $\max_{j \in N}\{a_j\} \le W \le r_{\max}$. The specific value of W can be set by an expert. Alternatively, we suggest to set $W = \min\{r_{\max}, \max\{\max_{j \in N}\{a_j\}, \sum_{j \in N^+}\{a_j\}\}$, where N^+ is the set of vertices of the precedence graph G which have no predecessor.

Set $r_j = a_j$ and calculate $p_j(r_j)$ for $j = 1, \ldots, n$. Initiate *ready times* T_i *of workers*: $T_i = 0$, $i = 1, \ldots, W$, and *ready times* t_j *of operations*: $t_j := 0$, $j = 1, \ldots, n$.

Step 2 (Set N^+) In graph G, calculate set N^+ of operations having no predecessor.

Step 3 (Assignment of operations) Select $j^* \in N^+$ with the largest value $p_j(r_j)$. Assign operation j^* to r_{j^*} consecutively numbered workers out of W workers such that all of them start performing this operation at the same earliest time, but not earlier than the operation ready time t_{j^*} and the ready times T_i of these workers. Update ready time of each such worker i: $T_i := T_i + p_{j^*}(r_{j^*})$. Update ready time of every immediate successor j of vertex j^*: $t_j := T_i$, where worker i is used to perform operation j^*. Update set N^+ by removing vertex j^* from it: $N^+ := N^+ \backslash \{j^*\}$. If set N^+ is empty, then remove its vertices and their outgoing arcs from G. If G is empty, then a complete schedule is constructed. In this case, perform computations in paragraphs (i)-(iii).

(i) Calculate its makespan $C_{\max} = \max\{T_i \mid i = 1, \ldots, W\}$.

(ii) If $C_{\max} \le d$, then output the corresponding schedule with W workers and stop.

(iii) Let $C_{\max} > d$. If $W = r_{\max}$, then output the infeasible solution with $r_{\max}$ workers and stop. If $W \le r_{\max} - 1$, then determine operations of a *critical path*, which do not intersect in time and whose processing times sum up to $C_{\max}$. Re-set $r_j := \min\{r_j+1, b_j, W+1\}$ for every such operation, re-set $W := W + 1$, restore original graph G, and go to Step 2.

If G is not empty, then go to Step 2. If set N^+ is not empty, then repeat Step 3. □

Heuristic $TopLong$ is run until a feasible solution is constructed, or an infeasible solution with $r_{\max}$ workers is found, or a given solution time limit is exceeded, in which case its output is an infeasible solution found last.

Our randomized heuristic $TopRandom$ differs from heuristic $TopLong$ in that in Step 3 operation $j^* \in N^+$ is selected at random. If the solution time limit permits, then heuristic $TopRandom$ can be run several times.

We remark that heuristics $TopLong$ and $TopRandom$ generate a solution, in which workers assigned to the same operation are numbered consecutively. We were recently aware of the results of Bladek et al. (2013) which imply that there are instances of the problem $MinNumber$ for which no optimal solution of this type exists.

4. REDUCTION TO A SERIES OF FEASIBILITY PROBLEMS

Consider a problem, which is to determine if there exists a feasible solution of the problem $MinNumber$ with a given number Q of workers. We denote this problem as $Feasible(Q)$. The problem $MinNumber$ can be solved by the following *bisection search* procedure over the range of possible values of Q.

Let LB be the number of workers such that the problem $MinNumber$ has no feasible solution, and let UB be the number of workers such that it has a feasible solution. Before the bisection search starts, we apply heuristics $TopLong$ and $TopRandom$. If the heuristics found no feasible solution, then we solve the problem $Feasible(r_{\max})$. If no feasible solution of this problem is found, then the problem $MinNumber$ has no feasible solution, and we stop. If a feasible solution is found, then we set $UB = r_{\max}$. If the heuristics found a feasible solution of the problem $MinNumber$, then we set UB to be the minimum value of the feasible solutions.

Then we solve the problem $Feasible(LB)$, where $LB := \max\{a_i \mid i \in N\}$. If a feasible solution of this problem is found, then it is an optimal solution of the problem $MinNumber$, and we stop. If no feasible solution is found, then the bisection search procedure starts. Its generic iteration can be described as follows.

If $UB = LB + 1$, then a feasible schedule for the problem $Feasible(UB)$ is an optimal schedule for the problem $MinNumber$. If $UB \geq LB + 2$, then solve the problem $Feasible(Q)$ for $Q = \lceil \frac{UB+LB}{2} \rceil$. If no feasible solution is found, then re-set $LB := Q$ and repeat the generic iteration. If a feasible solution is found, then re-set $UB := Q$ and repeat the generic iteration.

The number of iterations of the bisection search procedure is $O(\log_2(UB - LB))$, and in each iteration the problem $Feasible(Q)$ is solved for a $Q \in \{LB, LB+1, \ldots, UB\}$.

5. RELATION TO MULTI-MODE PROJECT SCHEDULING AND MULTIPROCESSOR SCHEDULING

The problem $Feasible(Q)$ can be formulated as a multi-mode project scheduling problem with multiple renewable resources (workers). The known mathematical programming formulations of such problems include variables with indices whose number is equal to the number of resources or variables with an index representing a mode, which is a subset of resources. Since workers is a non-homogeneous resource in the problem $Feasible(Q)$ in the sense that a solution should specify not only the number of workers to perform an operation, but also identify these workers, the number of resources is equal to r in our case, and we need variables with r indices to apply one of the earlier modeling approaches. By the same reason, there can be $2^Q - 1$ different modes and the same number of variables in the approach based on modes.

Most popular solution techniques for project scheduling problems include time-indexed and event-indexed MILP formulations, branch and bound schemes and meta-heuristics, see, for example, Monma et al. (1990), Demeulemeester et al. (1996), Salewski et al. (1997), Ranjbar et al. (2007), Li and Womer (2012), Besikci et al. (2013), and Ghoddousi et al. (2013).

Problem $Feasible(Q)$ is a generalization of the multiprocessor *moldable* task scheduling problem, in which the processing time of a computer task depends on the number of identical processors allocated to it, and this number cannot change during the task execution. Processors assigned to the same task are not specified. According to the survey of multiprocessor task scheduling problems of Drozdowski et al. (1996), a minimization counterpart of this problem is denoted as $P|spdp - lin - \delta_j, prec|C_{\max}$ if the task processing time is inversely proportional to the number of assigned processors, $p_j(r) = p_j/r$, and as $P|spdp - any, prec|C_{\max}$ if processing time is an arbitrary function of the number of processors.

The most popular solution techniques for multiprocessor task scheduling problems reported in the literature are heuristics with performance guarantees, see, for example, Wang and Cheng (1991, 1992), Choundhary et al. (1994), Srinivasa Prasanna and Musicus (1994).

6. MILP MODEL FOR THE PROBLEM $FEASIBLE(Q)$

Let us introduce the following variables.

- $x_{ir} \in \{0,1\}$, $i \in N$, $r = 1, \ldots, Q$: $x_{ir} = 1$ if operation i is assigned to worker r, and $x_{ir} = 0$, otherwise.
- $y_{ij} \in \{0,1\}$, $i \in N$, $j \in N$, $i \neq j$: $y_{ij} = 1$ if operation i completes before or at the start of operation j, and $y_{ij} = 0$, otherwise. Note that $y_{ij} = 0$ implies that either j completes before or at the start of i, or some parts of i and j are performed in parallel. In the latter case, i and j must be performed by different workers in a feasible solution.
- $z_{ir} \in \{0,1\}$, $i \in N$, $r = 1, \ldots, Q$: $z_{ir} = 1$ if operation i is performed by r workers, and $z_{ir} = 0$, otherwise.
- $S_{ir} \geq 0$, $i \in N$, $r = 1, \ldots, Q$: start time of operation i if it is processed by worker r, and any non-negative value otherwise.

Let I denote the set of vertex pairs (i,j) that are independent with respect to the precedence constraints:

$$I = \{(i,j) \mid i \in N, j \in N, i \neq j, (i,j) \notin U, (j,i) \notin U\}$$

and let M denote a sufficiently large positive number. Below we give a Mixed Integer Linear Programming (MILP) formulation of the problem $Feasible(Q)$, for which we keep the same notation $Feasible(Q)$.

Problem $Feasible(Q)$:

$$S_{ir} + \sum_{q=1}^{Q} p_i(q) z_{iq} \leq d, \quad i \in N, \ r = 1, \ldots, Q, \qquad (1)$$

$$\sum_{r=1}^{Q} z_{ir} = 1, \quad i \in N, \qquad (2)$$

$$\sum_{r=1}^{Q} x_{ir} = \sum_{r=1}^{Q} r z_{ir}, \quad i \in N, \qquad (3)$$

$$y_{ij} = 1, \ y_{ji} = 0, \quad (i,j) \in U, \qquad (4)$$

$$S_{jh} - S_{iq} \geq \sum_{r=1}^{Q} p_i(r) z_{ir} - M(3 - y_{ij} - x_{iq} - x_{jh}),$$
$$i, j \in N, \ i \neq j, \ h, q = 1, \ldots, Q, \qquad (5)$$

$$y_{ij} + y_{ji} \leq 1, \quad (i,j) \in I, \qquad (6)$$

$$x_{ir} + x_{jr} - 1 \leq y_{ij} + y_{ji}, \ (i,j) \in I, \ r = 1, \ldots, Q, \qquad (7)$$

$$\sum_{r=1}^{Q} r z_{ir} \leq b_i, \quad i \in N, \qquad (8)$$

$$a_i \leq \sum_{r=1}^{Q} r z_{ir}, \quad i \in N, \tag{9}$$

$$x_{ir}, y_{ij}, z_{ir} \in \{0,1\}, \quad i,j \in N, \ r = 1,\dots,Q, \tag{10}$$

$$S_{ir} \geq 0, \quad i \in N, \ r = 1,\dots,Q. \tag{11}$$

In the problem $Feasible(Q)$, there are $(n^2 - n)/2 + 3nQ$ number of 0-1 variables and nQ real-valued variables S_{ir}. The number of constraints without the constraints specifying bounds on the variables or their exact values is equal to $Q^2(n^2 - n)/2 + n(Q + 4) + |I|(Q + 1)$.

The constraints (1) require that the solution must be feasible with respect to the cycle time d. The constraints (2) state that exactly one number of workers in the range $1,\dots,Q$ must be used for the execution of every operation. The constraints (3) ensure that if operation i is assigned some number of workers, say k, and, hence, $z_{ik} = 1$, then exactly k variables x_{ir} take value 1. The constraints (4) define $y_{ij} = 1$ and $y_{ji} = 0$ if operation i precedes operation j. The constraints (5) enforce worker q to complete operation i before or at the time when worker h starts operation j if it is decided that operation i completes before operation j starts, worker q performs operation i and worker h performs j, i.e., if $y_{ij} = 1$, $x_{iq} = 1$ and $x_{jh} = 1$. The constraints (6) ensure that, for independent operations i and j, the case when both $y_{ij} = 1$ and $y_{ji} = 1$ cannot happen. The constraints (7) guarantee that if two independent operations are assigned to the same worker, then one of these operations must be completed before or at the start time of the other, i.e., either $y_{ij} = 1$ or $y_{ji} = 1$ must hold. Assume that operations i and j are independent. Consider values of the pair (x_{ir}, x_{jr}). If $(x_{ir}, x_{jr}) = (1,1)$, i.e., worker r is assigned to i and j, then (6) and (7) guarantee that either $y_{ij} = 1$ or $y_{ji} = 1$ must take place, which, together with (5) for $h = q = r$, guarantees that worker r will not process parts of i and j simultaneously. If $(x_{ir}, x_{jr}) = (1,0)$, i.e., worker r is assigned to i and not to j, then $y_{ij} = 0$ and $y_{ji} = 0$ can happen, that is, processing of i and j can be done independently in time. Implication of the cases $(x_{ir}, x_{jr}) = (0,1)$ and $(x_{ir}, x_{jr}) = (0,0)$ is the same as for $(x_{ir}, x_{jr}) = (1,0)$. The constraints (8) and (9) effectuate the limits on the number of workers for each operation.

Note that the solution of the MILP problem $Feasible(Q)$ specifies the start times of the operations, their assignment to workers and the operation durations. This information is sufficient to construct the corresponding schedule.

7. CONCLUSIONS

This paper presents preliminary studies of the problem $MinNumber$, which is to schedule workers on a production line with workforce dependent operation processing times and precedence constraints between operations. The problem is proved NP-hard in the strong sense for several practical cases. Conventional and randomized heuristics are suggested and a reduction to a series of feasibility problems is described. We have shown that the feasibility problem has much in common with the multi-mode project scheduling problems and multiprocessor scheduling problems. We formulated this problem as a MILP problem. We are currently conducting computer experiments for evaluating the quality of the heuristics and the MILP model and for tuning them to provide better solution quality and improve running times.

REFERENCES

Araújo, F.F.B., Costa, A.M., Miralles, C., 2012. Two extensions for the ALWABP: Parallel stations and collaborative approach. International Journal of Production Economics 140(1), 483-495.

Belmokhtar, S., Dolgui, A., Guschinsky, N., Levin, G., 2006. Integer programming models for logical layout design of modular machining lines. Computers and Industrial Engineering 51(3), 502-518.

Besikci, U., Bilge, U., Ulusoy, G., 2013. Resource dedication problem in a multi-project environment, Flexible Services and Manufacturing Journal 25(1-2), 206-229.

Bladek, I., Drozdowski, M., Guinand, F., Schepler, X., 2013. On contiguous and non-contiguous parallel task scheduling, in submission.

Blum, C., Miralles, C., 2011. On solving the assembly line worker assignment and balancing problem via beam search, Computers & Operations Research 38(1), 328-329.

Borisovsky, P., Dolgui, A., Kovalev, S., 2012a. Modelling transfer line design problem via a set partitioning problem. Optimization Letters 6(5), 915-926.

Borisovsky, P., Dolgui, A., Kovalev, S., 2012b. Algorithms and implementation of a set partitioning approach for modular machining line design. Computers & Operations Research 39, 3147-3155.

Choundhary, A.N., Narahari, B., Nicol, D.M., Simha, R., 1994. Optimal processor assignment for a class of pipelined computations, IEEE Transactions on Parallel and Distributed Systems 5/4, 439-445.

Delorme, X., Dolgui, A., Kovalyov, M.Y., 2012. Combinatorial design of a minimum cost transfer line. Omega 40, 31-41.

Demeulemeester, E., Elmaghraby, S.E., Herroelen, W. 1996. Optimal procedures for the discrete time/cost tradeoff problem in project networks, European Journal of Operational Research, 88, 50-68.

Drozdowski, M., 1996. Scheduling multiprocessor tasks - An overview, European Journal of Operational Research 94, 215-230.

Du , J.,Leung J.Y-T., Young, G.H. 1991. Scheduling chain-structured tasks to minimize makespan and mean flow time, Information and Computation, 92(2), 219-236.

Garey, M.R., Johnson, D.S., 1979. Computers and intractability: A guide to the theory of NP-completeness. Freeman, San Francisco.

Ghoddousi, P., Eshtehardian, E., Jooybanpour, S., Javanmardi, A., 2013. Multi-mode resource-constrained discrete time-cost-resource optimization in project scheduling using non-dominated sorting genetic algorithm, Automation in Construction 30, 216-227.

Karabak, F., G uner, N.D., Satir, B., Kandiller, L., G ursoy, I., 2011. An optimization model for worker assignment of a mixed model vehicle production assembly line under worker mobility. In Proceedings of the 41st International Conference on Computers & Industrial Engineering, pp. 483-490.

Li, H., Womer, K., 2012. Optimizing the supply chain configuration for make-to-order manufacturing, European

Journal of Operational Research 221 (1), 118-128.

Moreira, M.C.O., Costa, A.M., 2013. Hybrid heuristics for planning job rotation schedules in assembly lines with heterogeneous workers. International Journal of Production Economics 141(2), 552-560.

Monma, C.L., Schrijver, A., Todd, M.J., Wei, V.K., 1990. Convex resource allocation problems on directed acyclic graphs: duality, complexity, special cases, and extensions, Mathematics of Operations Research 15, 736-748.

Mutlu, O., Polat, O., Supciller, A.A., 2013. An iterative genetic algorithm for the assembly line worker assignment and balancing problem of type II, Computers & Operations Research 40(1), 418-426.

Ranjbar, M.R., Kianfar, F., 2007. Solving the discrete time/resource trade-off problem in project scheduling with genetic algorithms Applied Mathematics and Computation 191 (2), 451-456.

Salewski, F., Schirmer, A., Drexl, A., 1997. Project scheduling under resource and mode identity constraints: Model, complexity, methods, and application, European Journal of Operational Research 102, 88-110.

Srinivasa Prasanna, G.N., Musicus, B.R., 1994. Generalized multiprocessor scheduling for directed acyclic graphs, in: Proceedings of Supercomputing 1994, IEEE Press, New York, 237-246.

Wang, Q., Cheng, K.H., 1991. List scheduling of parallel tasks, Information Processing Letters 37, 291-297.

Wang, Q., Cheng, K.H., 1992. A heuristic of scheduling parallel tasks and its analysis, SlAM Journal on Computing 21 (2), 281-294.