Decision Making in Manufacturing and Services
Vol. 1 • 2007 • No. 1–2 • pp. 91–110

Scheduling with High Variety of Customized Compound Products

Czesław Smutnicki*

Abstract. Domestic appliance is an instance of manufacturing various products on clients demand with frequent changes of production. Although the technological process for each individual product is relatively simply, the variety of products, mixed orders, frequent machines changeovers, machines with unusual service policy, lack or limited storage, etc., generates quite nontrivial planning, batching and scheduling problems and furthermore of a huge size. In this paper, we present specific real process of production of refrigerators, mathematical and graph models of the problem and an outline of solution algorithm, based the on local search approach.

Keywords: scheduling, tabu search.

Mathematics Subject Classification: 90B30, 90B35, 90C59.

Received/Revised: 29 March 2007/11 July 2007

1. INTRODUCTION

Optimization problems arising from industry, especially scheduling problems, pose a real challenge for the designing of efficient solution algorithms. Their NP-hardness and practical complexity disqualify classical, exact approaches, as an example, branch-and-bound schemes (B&B) or mixed integer linear programming (ILP, MILP), for sizes of instances that come from practice. On the other hand, the quality of solutions generated by approximate approaches influences immediately economic indexes (thus also profits of firms). That's why many researchers continuously seek new algorithms which would be able to solve practical scheduling problems with high accuracy in a short time.

Although many efforts have been made for fundamental and advanced scheduling models, the quality of solutions obtainable in a reasonable time remains not fully satisfactory. Vastly worse situation we have observed in the complex production systems, which generate troubles already with modelling, not saying anything just about effective solution methods. New modern techniques of production management (lean manufacturing, kanban, etc.) frequently have been proposed by experts as the universal "medicine" on planning and control ailments. One should be conscious that

* Wroclaw University of Technology, Institute of Computer Engineering, Control and Robotics, Poland. E-mail: czeslaw.smutnicki@pwr.wroc.pl.

these approaches, although very useful in practice, eliminate perfect optimal planning even in cases where mathematical optimization is the sole method for improvement of system activity.

In this paper, we discuss real problem of producing refrigerators, for which kanban technology has failed. Kanban failure in this case follows from the commonly known fact that in case of frequent changes of final products the put down amount of abandoned semi-products appears too heavy to carry it by the production system. Then, perfect planning of the production is especially welcome. Not concentrating of product mark or type, the developed methodology can be applied in firms producing washing machines, dishwashers, cookers, etc. in various market models and frequent changes of production profile. At first look, from the scheduling theory point of view, this problem could be perceived as either known in the literature *resource constrained project scheduling (RCPS) problem* or a generalization of *job-shop* problem toward the use of *multi-purpose* machines with *changeover* constraints. However, in *this* real case there exist simultaneously several untypical constraints that follow chiefly from the usage of forms (a kind of tool), the changeover policy and from the new class of carousel multi-channel machines. These imply that neither RCPS nor job-shop models are immediately applicable. On the other hand, complex MILP models with many hard constraints are commonly replaced by job shop like models, even in complex chemical processes, (Neumann et. al, 2003). This suggests that we should look for solution technology rather in conventional combinatorial scheduling approach than in MILP based approaches.

As to RCPS models and tools, we consciously skip the overview of the rich literature and only refer the reader to the excellent state of the art survey papers and analysis (Brucker et. al, 1999; Herroelen et. al 1996; Kolisch, Hartmann, 2001; Kolisch, Padman, 2001). In RCPS area, one can distinguish several research streams which follow from type of resource used (renewable, partially renewable, non-renewable, doubly constrained), type of relations between activity, number of criteria (single-, multiple-criteria), particular form of criteria, type of constraints and optimization directions. Applied solution methods strongly depends on the problem, stream, and theirs special properties, we mention only typical approaches (Brucker, 1998; Kolisch, Drexl 1997; Mori, Tseng, 1997; Özdamar, Ulusoy, 1996; Thomas, Salhi, 1998) but a few. In terms of RCPS, problem considered by us can be perceived as certain special class of multi-mode scheduling with renewable resources, (Salewski et. al, 1997; Sprecher, Drexel, 1998), changeover times and single time criteria. Unfortunately, taking into account the real size of instances, RCPS offers no powerful solution methods beside priority rules or tabu search, (Sprecher, Drexel, 1998; Thomas, Salhi, 1998). Hence, our basic aim is to propose the model of the described manufacturing process (taking into account all specific constraints) and the solution algorithm of different types, constructive, improvement type, TS type, (Glover, Laguna, 1996; Laguna, Marti, 2003). Significant attention has been paid for modelling process, in order to get link with graph models known in job-shop and specific RCPS scheduling, (Nowicki, Smutnicki, 1996), block approach existed for problems with min-max type criterion and excellent properties of well-known TSAB algorithm, (Nowicki, Smutnicki 1996, 2005a, 2005b).

The paper is organized as follows. After a short presentation of the problem, notions and denotations (Section 2) we introduce formal mathematical model with decision variables and constraints (Section 3). Batching is introduced in Section 4 as independent outer procedure responsible for setting the batch number and batch sizes, which can be perceived as the independent module of preparing input data for the scheduling problem. Following the mathematical model we introduce the auxiliary graph model (Section 5), which next has been built into multilevel algorithm obtained through the decomposition of the problem stated. Special properties of the problem, simplification and extensions in the context of industrial practice, as well as solution algorithm are presented in Section 6. Conclusions and future research directions are discussed in Section 8.

2. THE PROBLEM

In this section we introduce the real problem as well as all necessary notions and denotations, see also Figure 1. The problem is formulated as follows.

The factory can produce n different products on clients demand and let $N = \{1, 2, \ldots, n\}$ denote the set of these products. In fact, the real problem is much more complicated because each product can be provided on the market in various *models*, which basically differ each other by casing colour, shape and equipment. Hopefully, the technological staff have distinguished the minimal set of mutually different *logistic products* which do not depend on casing parameters, so we can calmly assume that N contains only essentially various products. Let n_i denote the demand size of product i ordered by a client, having the supply date d_i, $i \in N$.

Without losing generality, we assume that $n_i > 0$, $i \in N$, otherwise we can reduce set N appropriately. Copy of product i needs for processing the set of operations linked by certain graph, see for example Figure 2. Let O_i and $G_i = (O_i, E_i)$, where $E_i \subset O_i \times O_i$, denote respectively the *set of operations* and the *graph of technological order of operations* in activity-on-node notation for product i. We assume next that all operations own unique indexes, so let $O = \bigcup_{i \in N} O_i$ and $G = (O, E)$, where $E = \bigcup_{i \in N} E_i$. Each operation $j \in O$ needs for processing various resources, namely: single *machine* $a \in U_j$ (selected from the *given subset of machines* U_j) and the *fixed subset of forms* F_j. We treat n_i copies of product i as the separate batch, which means that operation j corresponds, in fact, to processing n_i identical activities successively. Forms play the crucial role in the production process, as it will be discuss later, since different products may have the

same structure of the operation graph, but through using completely different set of forms provide different product.

There are three types of machines, each of them having single human operator:

(1) sequential, which can process at most one operation at the time,
(2) multi-channel, which can process simultaneously and independently several parts at the time,
(3) carousel, which can process simultaneously

several parts (operations are delayed successively by a constant in a cyclic way), whereas only one part is accessible at a time, see Figure 3.

Let M denotes the set of machines of all types, $M = M^s \cup M^m \cup M^c$, where $M^s = \{1, 2, \ldots, m_s\}$, $M^m = \{m_s + 1, m_s + 2, \ldots, m_m\}$, $M^c = \{m_m + 1, m_m + 2, \ldots, m_c\}$ denote sets of sequential, multi-channel and carousel machines, respectively.

notion	meaning
$N = \{1, 2 \ldots, n\}$	set of products
i	product index
O_i	set of operations for product i
$E_i \subset O_i \times O_i$	technological order of operations from O_i
$G_i = (O_i, E_i)$	graph of precedence constraints for product i in activity-on-node notation
d_i	supply date for product i
$O = \bigcup_{i=1}^{n} O_i$	set of all operations
$o = \|O\|$	total number of operations
$E = \bigcup_{i=1}^{n} E_i$	set of arcs for graph $G = (O, E)$
j	operation index
a	machine index
U_j	set of machines required alternatively by operation $j \in O$
F_j	set of form types required simultaneously by operation $j \in O$
n_i	size of demand for product i
$M = M^s \cup M^m \cup M^c$	set of machines
$M^s = \{1, 2, \ldots, m_s\}$	set of sequential machines
$M^m = \{m_s + 1, \ldots, m_m\}$	set of multichannel machines
$M^c = \{m_m + 1, \ldots, m_c\}$	set of carousel machines
k_a	the number of channels in multichannel machine $a \in M^m$
(a, c)	(machine, channel), $c \in \{1, 2, \ldots, k_a\}$, $a \in M^m$
p^*_{aj}	processing time of operation j on machine a
$p_{aj} = n_i p^*_{aj}$	processing time of aggregated operation j being the batch of size n_i
$H = \{1, 2, \ldots, h\}$	set of form types
b	form type index
h_b	the number of forms of type b
(b, c)	(form type, form index), $c \in \{1, 2, \ldots, h_b\}$, $b \in H$
$s(a, F_x, F_y)$	time necessary to make changeover on machine a from form set F_x to F_y

variable	meaning
S_j, C_j	starting, completion time of operation $j \in O$
R_j	machine allocated for operation $j \in O$
Q_j	set of forms allocated for operation $j \in O$
$P_F(Q_j)$	projection of Q_j onto F
(R, Q, S, C)	schedule, $R = (R_1, \ldots, R_o)$, $Q = (Q_1, \ldots, Q_o)$, $S = (S_1, \ldots, S_o)$, etc.
b_{ij}	size of batch for product i and operation j
W_R	set of operations being in machine conflict, see (27)
W_Q	set of operations being in form conflict, see (28)

Fig. 1. List of notions used in the paper

Thus $U_j \subset M$, more precisely U_j is included in the whole in exactly one of subsets M^s, M^m, M^c. Each composed machine a of type (2) and (3) can be considered as a set (bank) of k_a identical uniform parallel machines, $a \in M^m \cup M^c$.

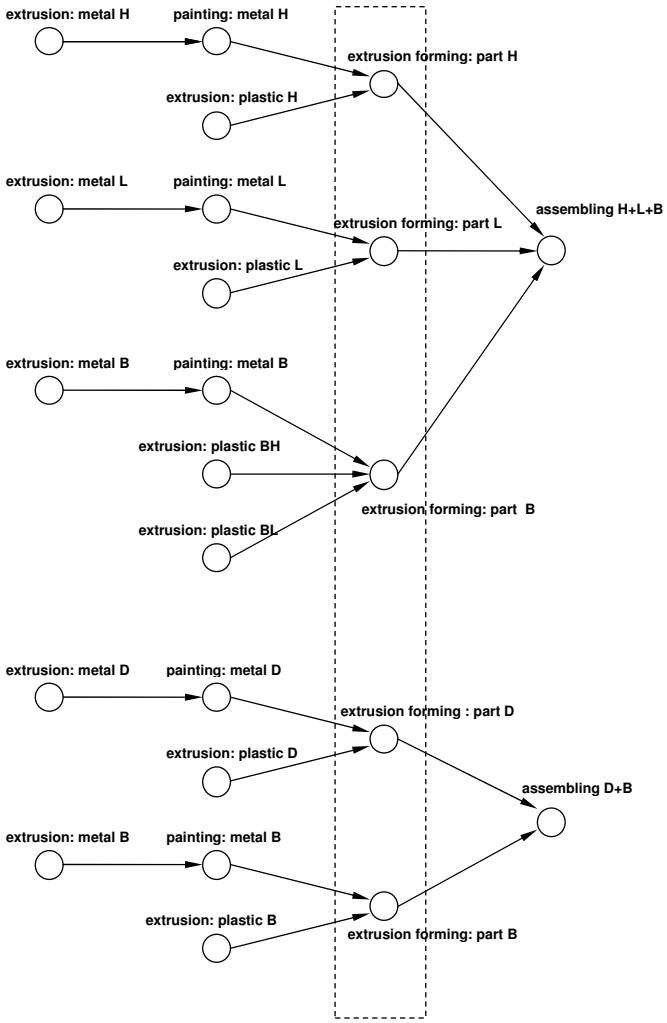

Fig. 2. Technological order (graph G_i) of operations for two typical products: refrigerator with freezer (upper) and refrigerator alone (lower)

Specified machine from the bank can process an operation j if and only if all forms F_j required by this operation have been mounted on this machine.

From the operation point of view there is no meaning which particular machine services the operation, however this is significant for forms management and changeover. Therefore, in the sequel, machines will be identified by pairs (a, c), where $c \in \{1, 2, \ldots, k_a\}$, $a \in M$, since forms are mounted on the particular machine c in the bank a.

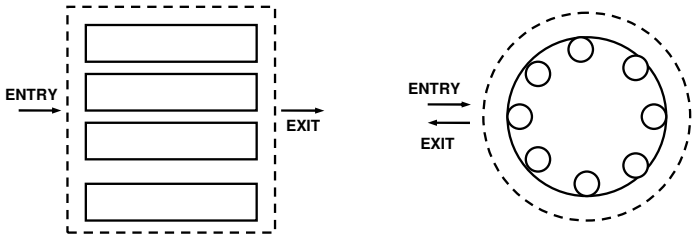

Fig. 3. Multi-channel machine (left) and carousel machine (right)

For symmetry we set $k_a = 1$ for $a \in M^s$. Processing time of operation j (for single copy of product i) processed on machine $a \in U_j$ is equal p^*_{aj}, $j \in O_i$, $i \in N$. The composed batch-operation has real processing time $n_i p^*_{aj}$, $j \in O_i$, $i \in N$, however we use hereinafter abbreviation p_{aj} meaning in fact the term $n_i p^*_{aj}$.

Forms are dedicated for specified products, however different products may use the non-disjoin sets of forms. We denote by $H = \{1, 2, \ldots, h\}$ the set of form types, and by h_b the number of forms of type $b \in H$ available in the system. Next we denote by $s(a, F_x, F_y)$ the time necessary to make changeover on machine $a \in M$ from set form $F_x \subset F$ to set form $F_y \subset F$. For symmetry we set $s(a, F_x, F_y) = 0$ if $F_x = F_y$ for any a, and define additionally notions of initial changeover $s(a, \emptyset, F_y)$ and final changeover $s(a, F_x, \emptyset)$. Composite machines of types (2) and (3) differ fundamentally each other by the technology of making changeover. In particular, carousel machines has not been considered yet in the literature at all. The changeover of forms on a single machine (a, c) in the bank $a \in M^m$ does not have any influence on the activity of other machines in this bank as well as on machines in other banks from M^m, see Figure 4.

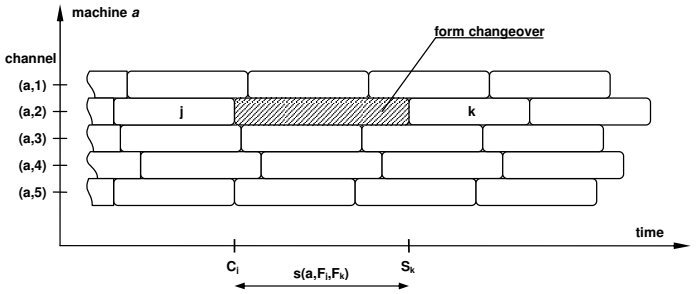

Fig. 4. Multi-channel machine. Form changeover policy. Basic case

The changeover of forms on a single machine (a, c) in the bank $a \in M^c$ stops the whole bank a (this carousel) for the time of making setup, see Figure 5. Since rules of form changeover are a bit complex, similarly as for machines, forms will be denoted by pairs (b, c), where $c \in \{1, 2, \ldots, h_b\}$, $b \in H$.

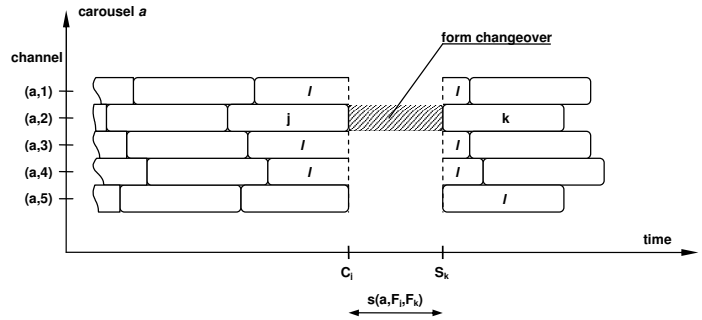

Fig. 5. Carousel machine. Form changeover policy. Basic case

Now, we are ready to formulate solution, constraints and the optimization criteria. Let S_j and C_j denote the decision variables being *starting time* and *completion time* of operation j. We consider machines and forms as renewable resources R_j, Q_j allocable for operation j. Let R_j be the decision variable denoting *machine allocated for operation* j. Then we have $R_j = (a, c)$, where $c \in \{1, 2, \ldots, k_a\}$, $a \in U_j \subset M$, $j \in O$. Let Q_j be a decision variable denoting *set of forms allocated for operation* j. Then we have $Q_j = \{(b_1, c_1), (b_2, c_2), \ldots, (b_q, c_q)\}$, where $P_F(Q_j) \stackrel{\text{def}}{=} \{b_1, b_2, \ldots, b_q\} = F_j$, and $c_t \in \{1, 2, \ldots, h_{b_t}\}$, $t = 1, 2, \ldots, q$, $q = |F_j|$. $P_F(Q_j)$ is the projection of set Q_j onto F done by choosing prime components of pairs from Q_j. The schedule is represented by quadruple (R, Q, S, C), where $R = (R_1, R_2, \ldots, R_o)$, $Q = (Q_1, Q_2, \ldots, Q_o)$, $S = (S_1, S_2, \ldots, S_o)$, $C = (C_1, C_2, \ldots, C_o)$, $o = |O|$. After consultation, experts from the factory suggest to use as the optimization criteria either *makespan* or maximum *lateness*, which we would like to minimize. Since the solution methodology in both cases is the same, whereas the latter criteria is slightly more general, we decided to use lateness hereinafter.

3. MATHEMATICAL MODEL

This section aim is to formulate the formal mathematical model of the problem. It provides the framework for the solution method as well as foundations for the graph model employed in the algorithm. The goal function value, given below, represents the lateness, whereas constraints are listed in the further part of this section

$$\min_{R,Q,S,C} \quad \max_{j \in O} \ (C_j - d_j). \tag{1}$$

Condition (2) allows only feasible allocations of machines to operation and ensures that capacity of machines from $M^m \cup M^c$ won't be violated,

$$R_j = (a, c), \quad c \in \{1, 2, \ldots, k_a\}, \quad a \in U_j \subset M, \quad j \in O. \tag{2}$$

Condition (3) ensures that required forms for operation will be allocated,

$$P_F(Q_j) = F_j, \quad j \in O. \tag{3}$$

Condition (4) ensures that available number of forms won't be exceeded,

$$Q_j = \{(b_1, c_1), (b_2, c_2), \ldots, (b_q, c_q)\},\ c_t \in \{1, 2, \ldots, h_{b_t}\},\ t = 1, 2, \ldots, q = |F_j|\,,\ j \in O. \tag{4}$$

Condition (5) naturally limits starting times,

$$S_j \geq 0,\ \ j \in O. \tag{5}$$

Condition (6) forces the technological order of operations in products,

$$C_j \leq S_k,\ \ (j,k) \in E_i,\ \ i \in N. \tag{6}$$

Condition (7) takes into account processing time of operation for the chosen machine $R_j = (a, c)$ allocated for this operation,

$$S_j + p_{aj} \leq C_j,\ \ j \in O. \tag{7}$$

Changeover constraints depend on machine type and type of resource conflict. If two operations $k, j \in O$ use the same machine $R_j = (a, c) = R_k$, where $a \in M^s \cup M^m$, then we have disjunctive condition (symbol $\dot\vee$ means disjunction) with suitable changeover times, see also Figure 4,

$$(C_j + s(a, F_j, F_k) \leq S_k)\ \dot\vee\ (C_k + s(a, F_k, F_j) \leq S_j). \tag{8}$$

Notice, by the definition $s(a, F_j, F_k) = 0$ if $F_j = F_k$. If two operations $k, j \in O$ use different machines $(a, c) = R_j \neq R_k = (a', c')$, where $a, a' \in M^s \cup M^m$, but have conflicted set of forms $Q_j \cap Q_k \neq \emptyset$ then we have disjunctive condition with a extended changeover times, being the sum of times of mount off and on proper forms, namely

$$(C_j + s(a, F_j, \emptyset) + s(a', \emptyset, F_k) \leq S_k)\ \dot\vee\ (C_k + s(a', F_k, \emptyset) + s(a, \emptyset, F_j) \leq S_j). \tag{9}$$

If two operations $k, j \in O$ use the same machine $R_j = (a, c) = R_k$, where $a \in M^c$, then we have disjunctive set of conditions "either (10) or (13)", completed by auxiliary conditions (11)–(12) or (14)–(15), respectively,

$$C_j + s(a, F_j, F_k) \leq S_k. \tag{10}$$

Taking (10) we have to add simultaneously conditions that force temporary stop carousel a in the interval $[C_j, C_j + s(a, F_j, F_k)]$, namely

$$S_l + p_{al} + s(a, F_j, F_k) \leq C_l, \tag{11}$$

for any l such that $R_l = (a, e)$, $e \in \{1, 2, \ldots, k_a\}$, $e \neq c$, $S_l \leq C_j < S_l + p_{al}$ and also

$$C_j + s(a, F_j, F_k) \leq S_l \tag{12}$$

for any l such that $R_l = (a, e)$, $e \in \{1, 2, \ldots, k_a\}$, $e \neq c$, $C_j < S_l < C_j + s(a, F_j, F_k)$. By analogy, complementary to (10) condition

$$C_k + s(a, F_k, F_j) \leq S_j, \tag{13}$$

we have to supplement by analogous conditions stopping carousel a

$$S_l + p_{al} + s(a, F_k, F_j) \leq C_l, \tag{14}$$

for any l such that $R_l = (a, e)$, $e \in \{1, 2, \ldots, k_a\}$, $e \neq c$, $S_l \leq C_k < S_l + p_{al}$ and also

$$C_k + s(a, F_k, F_j) \leq S_l \tag{15}$$

for any l such that $R_l = (a, e)$, $e \in \{1, 2, \ldots, k_a\}$, $e \neq c$, $C_k < S_l < C_k + s(a, F_k, F_j)$.

If two operations $k, j \in O$ use different machines $(a, c) = R_j \neq R_k = (a', c')$, where $a, a' \in M^c$, but have conflicted set of forms $Q_j \cap Q_k \neq \emptyset$ then we have disjunctive conditions "either (16) or (21)", completed by auxiliary conditions (17)–(20) or (22)–(25) respectively,

$$C_j + s(a, F_j, \emptyset) + s(a', \emptyset, F_k) \leq S_k. \tag{16}$$

Taking (16) we have to add simultaneously conditions that force temporary stop carousel a in the interval $[C_j, C_j + s(a, F_j, \emptyset)]$, namely

$$S_l + p_{al} + s(a, F_j, \emptyset) \leq C_l, \tag{17}$$

for any l such that $R_l = (a, e)$, $e \in \{1, 2, \ldots, k_a\}$, $e \neq c$, $S_l \leq C_j < S_l + p_{al}$, and also

$$C_j + s(a, F_j, \emptyset) \leq S_l \tag{18}$$

for any l such that $R_l = (a, e)$, $e \in \{1, 2, \ldots, k_a\}$, $e \neq c$, $C_j < S_l < C_j + s(a, F_j, \emptyset)$. Moreover carousel a' has to be stopped in the interval $[C_j + s(a, F_j, \emptyset), C_j + s(a, F_j, \emptyset) + s(a', \emptyset, F_k)]$, so we need to add conditions

$$S_l + p_{a'l} + s(a', \emptyset, F_k) \leq C_l, \tag{19}$$

for any l such that $R_l = (a', e)$, $e \in \{1, 2, \ldots, k_{a'}\}$, $e \neq c'$, $S_l \leq C_j + s(a, F_j, \emptyset) < S_l + p_{a'l}$, and also

$$C_j + s(a, F_j, \emptyset) + s(a', \emptyset, F_k) \leq S_l \tag{20}$$

for any l such that $R_l = (a', e)$, $e \in \{1, 2, \ldots, k_{a'}\}$, $e \neq c'$, $C_j + s(a, F_j, \emptyset) < S_l < C_j + s(a, F_j, \emptyset) + s(a', \emptyset, F_k)$. By analogy, complementary to (16) condition

$$C_k + s(a', F_k, \emptyset) + s(a, \emptyset, F_j) \leq S_j \tag{21}$$

we have to supplement by conditions that stopped carousel a' in the interval $[C_k, C_k + s(a', F_k, \emptyset)]$

$$S_l + p_{a'l} + s(a', F_k, \emptyset) \leq C_l, \tag{22}$$

for any l such that $R_l = (a', e)$, $e \in \{1, 2, \ldots, k_{a'}\}$, $e \neq c'$, $S_l \leq C_k < S_l + p_{a'l}$ and also

$$C_k + s(a', F_k, \emptyset) \leq S_l \tag{23}$$

for any l such that $R_l = (a', e)$, $e \in \{1, 2, \ldots, k_{a'}\}$, $e \neq c'$, $C_k < S_l < C_k + s(a', F_k, \emptyset)$. Moreover carousel a has to be stopped in the interval $[C_k + s(a', F_k, \emptyset), C_k + s(a', F_k, \emptyset) + s(a, \emptyset, F_j)]$, so we need to add conditions

$$S_l + p_{al} + s(a, \emptyset, F_j) \leq C_l, \tag{24}$$

for any l such that $R_l = (a, e)$, $e \in \{1, 2, \ldots, k_a\}$, $e \neq c$, $S_l \leq C_k + s(a', F_k, \emptyset) < S_l + p_{al}$, and also

$$C_j + s(a', F_k, \emptyset) + s(a, \emptyset, F_j) \leq S_l \tag{25}$$

for any l such that $R_l = (a, e)$, $e \in \{1, 2, \ldots, k_a\}$, $e \neq c$, $C_k + s(a', F_k, \emptyset) < S_l < C_k + s(a', F_k, \emptyset) + s(a, \emptyset, F_j)$.

The model presented above does not reflect complex reality in all details, however we consciously omit a lot of them chiefly in order to simplify description and analysis. How complex is the problem (1)–(25) one can imagine providing real instance data. We get from the industry: $n \approx 50$, $\sum_{s=1}^{n} n_i \approx 5000$ (plan per week), $|F| = f = 50$, $\sum_{t=1}^{f} h_t \approx 300$, $|M| \approx 25$, $|M^m| \approx 4$, $\sum_{t=m_s+1}^{m_m} k_t \approx 30$, $|M^c| \approx 5$, $\sum_{t=m_m+1}^{m_c} k_t \approx 70$, etc. Planning horizon gather three periods: (1) 2–3 recent days where schedule theoretically cannot be changed (in practice few "very important" changes usually occur), (2) w week – typical planning horizon with three shifts per day, (3) a month – inflow of orders collected for pre-planning. The great number of logistic models implies frequent changeovers – at least a few per shift for carousel machines and few for multichannel machines. Utilization of machines is about 80-90%. Processing times for various models change in the range 20%. It is clear that transformation (1)–(25) into MILP will not fulfill our expectations regarding solution methods.

The manufacturing process is organized in specific way. Operations, see Figure 2, called "extrusion: metal", "painting: metal" and "extrusion: plastic" are performed using machines from M^s. For these operations forms are associated with each particular machine, which implies that problem of form allocation for these operations does not exist. Operations "extrusion forming: part X", where $X \in \{H, L, D\}$ are performed using machines from M^c. Each such operation requires typically a couple of forms. One can introduce certain simplification, by allocation only pre-defined couples of forms, which reduces the number of considered variants of allocation. Operations "extrusion forming: part B" are performed using machines from M^m. Each such operation requires from two to five forms, different than those for previously mentioned operations. Assembling operations are performed on the assembly line, no additional resources are required.

4. BATCHING

In this section we discuss possibility of embedding portions of the order (used by the planning department of the factory) in the formulated already mathematical model. It is clear that by performing order i sequentially, in the whole, we prevent unnecessary machine changeovers, so we can reduce costs and waste time. Moreover, a quite small number of forms is required. One can say that this strategy should be preferred. On

the other hand, there are several significant reasons which justify splitting the order i of size n_i into smaller portions called *batches*. The first argument follows from the higher capacity of successive stage (usually the assembly line) which forces processing parts from order i simultaneously on several parallel machines preceding the assembly line. The second argument is justified for orders of large size from single client – in fact, this order will be shipped in portions, because of the transport property. This approach ensures smoother policy of supplying as the side effect.

Allowing batching, we assume that order i of size n_i is split into w_i batches of non-zero sizes $b_{i1}, b_{i2}, \ldots, b_{iw_i}$, $\sum_{j=1}^{w_i} b_{ij} = n_i$, $i \in M$. Then, product i should be rather denoted by set of pairs (i, c), where $c \in \{1, 2, \ldots, w_i\}$ is the index of batch; all such batches of product i own the same due date d_i, $i \in N$. Similarly, operation j should be written rather as pairs (j, c), where $c \in \{1, 2, \ldots, w_i\}$ is the index of batch. Processing time of such composed batch-operation can be found as $b_{ic}p^*_{kj}$, where p^*_{kj} is the processing time for the unit of product i. So we need to re-define appropriately all necessary notions, namely set of tasks, set of operations and graph. This will complicate description of the problem as well as presentation of the algorithm. Therefore, we do not do it chiefly for the simplicity of notation. We assume hereinafter that: (A) batching has been already made and the *extended set of products* N^* consists portions of repetitive products in form of batches of sizes denoted by n_i instead of b_{ij}, (B) each batch-operation j has processing time equal $n_i p^*_{kj}$, where $k \in U_j$, $j \in O_i$.

Open remains the policy of selecting batch size. It results from either optimization process or firm policy. Exploiting rules currently used in the firm, following strategies have been proposed, implemented and tested in the algorithm:

(1) constant batch size b^*, where b^* is a parameter, i.e. $w_i = \lceil n_i/b^* \rceil$, $b_{ij} = b^*$, $j = 1, \ldots, w_i - 1$, $b_{iw_i} = n_i - B(w_i - 1)$,
(2) proportional batch size, i.e. $w_i = n^*$, where n^* is a parameter, $b_{ij} = \lfloor n_i/n^* \rfloor$, $j = 1, \ldots, w_i - 1$, $b_{iw_i} = n_i - \sum_{j=1}^{w_i - 1} b_{ij}$,
(3) constant (or proportional) batch size for large orders only, i.e. for these i such that $n_i > n^{**}$; small orders are not spliced onto batches.

Notice that b^*, n^*, n^{**} can be used as control parameters in an independent outer optimization procedure, which provides modified input data set.

5. GRAPH MODEL

In this section we provide foundations for the solution algorithm, in particular we introduce auxiliary graph model used in the most internal part of the method. Let us assume that splitting jobs into batches has been already done, providing expanded set of products N^* which implies suitable set of operations O. Then, optimization (1) can be organized in the following hierarchical way

$$\min_{R,Q} \ \min_{S,C} \ \max_{j \in O} \ (C_j - d_j). \tag{26}$$

Let R, Q be fixed but feasible in the sense of constraints (2)–(4). Then, decision variables S, C depend, among others, on the choice made between alternatives in all

disjunctive conditions (8)–(10), (13), (16), (21), each of which fix precedence either $j \to k$ or $k \to j$ for operations j, k being in resource conflict, namely in *machine conflict*

$$W_R = \{(j,k),(k,j): \ \ R_j = R_k, \ \ j,k \in O\} \tag{27}$$

and in *form conflict*

$$W_Q = \{(j,k),(k,j): \ \ R_j \neq R_k, \ \ Q_j \cap Q_k \neq \emptyset, \ \ j,k \in O\} \tag{28}$$

Assume next that exactly *single* pair from each two disjunctive pairs $(j,k),(k,j)$ has been chosen, providing *representations of disjunctions* $T_R \subset W_R$ and $T_Q \subset W_Q$. This means that problem (26) has been additionally decomposed by introducing T_R, T_Q, namely

$$\min_{R,Q} \ \min_{T_R,T_Q} \ \min_{S,C} \ \max_{j \in O} \ (C_j - d_j). \tag{29}$$

Following (29) we propose hierarchical algorithm which carries out appropriate minimization in layers, see Figure 6. On the lowest level we need to solve the problem

$$\min_{S,C} \ \max_{j \in O} \ (C_j - d_j). \tag{30}$$

for fixed R, Q, T_R, T_Q. To this aim, we construct graph $G(R,Q,T_R,T_Q) = (O, E \cup T_R \cup T_Q)$, where O and E follow from the graph of technology order $G = (O,E)$ defined in Section 2. Node $j \in O$ represents operation j and has associated two events: starting time S_j and completion time C_j. Node j has weight p_{aj}, where $a = R_j$. Arc $(j,k) \in E$ has weight zero. Remain arcs have weights that depend on disjunctive conditions set in the mathematical model. Arc $(j,k) \in T_R$ has weight $s(a, F_j, F_k)$, where $R_j = a = R_k$, see (8) and (10) or (13). Arc $(j,k) \in T_Q$ has weight $s(a, F_j, \emptyset) + s(a', \emptyset, F_k)$, where $R_j = a \neq a' = R_k$, see (9) and (16) or (21). Observe that conditions (11)–(12) (appropriately (14)–(15)) and conditions (17)–(20) (appropriately (22)–(25)) are still not respected in the graph $G(R,Q,T_R,T_Q)$, because cannot be expressed using fixed a priori graph arcs. Therefore we propose special procedure of finding S, C for (30) which consists of two phases: (A) generating and (B) improvement. In the former phase constraints (11)–(12), (14)–(15), (17)–(20), (22)–(25) are relaxed and then S_j, C_j are found through longest paths in the graph, however so found values are only lower bounds instead of real starting and completion times. In the latter phase relaxed conditions are introduced step-by-step, modifying S_j, C_j toward feasible ones in sense of the whole model (1)–(25). Let us consider these phases in detail.

Phase (A) can be performed if and only if graph $G(R,Q,T_R,T_Q)$ does not contain cycle. Then, starting time S_j of operation j is the length of the longest path going to the node j, without the weight of this node. Completion time C_j of operation j equals S_j plus the weight of node j. Using commonly known transformation, we convert maximum lateness value into critical path length in standard problem of finding critical path in the graph. At the begin of phase (B) we sort all events S_j into nondecreasing order. Our aim is to check events, along this order, for possible violation of previously relaxed constraints (11)–(12), (14)–(15), (17)–(20),

(22)–(25). For example, if we found for some j that $(j,k) \in T_R$, $a = R_j \in M^c$ and $s(a, F_j, F_k) > 0$, then we modify processing time p_{al} (weight of node l) by adding term $s(a, F_j, F_k)$, see (11), for all nodes l such that $R_l = (a, e)$, $e \in \{1, 2, \ldots, k_a\}$, $e \neq c$, $S_l \leq C_j < S_l + p_{al}$. Additionally, for any node l such that $R_l = (a, e)$, $e \in \{1, 2, \ldots, k_a\}$, $e \neq c$, $C_j < S_l < C_j + s(a, F_j, F_k)$ we introduce additional arc (j, l) with weight $s(a, F_j, F_k)$, see (12). Such modification influences on graph structure (provides its temporary modification), thus can change values of S_j, C_j. Therefore, immediately after modification we need to run critical path method once again, actually for some part of the graph. We repeat phase (B) until no changes have been introduced. Retrieving other types of feasibility is made in analogous way. The described procedure provides not necessary optimal values for the problem (30), however in experiments it is very close to optimal.

6. ALGORITHM

Process administrator has proven that currently bottleneck of the production system occurs on machines from M^m and M^c. At the analysis described below we accept this point of view. Following SQUEZEE philosophy we use OPT management strategy. In OPT we relax all but bottleneck stage, then optimally schedule the bottleneck stage and at the end "spread" solution to obtain feasible schedule of remain stages. Thus, we resign from the scheduling on machines M^s as well as the assembly line balancing, and consider only operations processed on machines M^m and M^c, see area bounded by dashed line in Figure 1.

Without the loss of generality we assume that:

(a) these operations are numbered consecutively $1, 2 \ldots, o$,
(b) operation j can be processed using machine $a \in U_j$ and set of forms F_j, as it has been already defined,
(c) obligatory are changeover rules set previously.

Following decomposition in (26) we propose the hierarchical algorithm shown in Figure 6. The lowest level has been already proposed and discussed in detail in Section 5.

Let us go to successive higher levels. Next two levels, namely $\min_{R,Q}$ and $\min_{T_R,T_Q}$, can be considered either separately or jointly. The former approach is more suitable in the context of decomposition already made. It also allow us to isolate loading from sequencing in their influence on the final schedule as well as to use different optimization techniques on these levels. Many authors do so, not discussing neighbourhood size derived from this approach in the context of local search methods. On the other hand, joint treatment of loading and sequencing leads us to certain homogeneity in defining and considering moves for local search needs, see e.g. TS (Nowicki, Smutnicki 1998), which results in faster convergence to good solution as well as prevents explosion of the neighbourhood size and search cost in local search approaches.

Joint optimization can be realized in several ways:

(a) outline of the load and sequence, found by certain optimization method, spread then into detailed schedule,
(b) common moves for loading and sequencing, see (Nowicki, Smutnicki 1998),
(c) priority list algorithm.

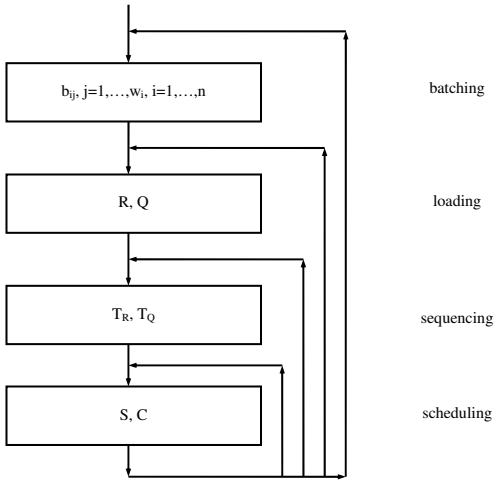

Fig. 6. Multi-level solution algorithm

Let us consider these approaches in the context of the problem stated.

Implementation of (a) bring to large size optimization tasks, its applicability needs further studies. Implementation of (b) implies complex forms of moves (because of complex resource requirements for single operation) and large size of the neighbourhood (because of the real instance size). Although the former disadvantage (complex moves) has been partially resolved, still open remains efficient method of the neighbourhood reduction. Not abandoning completely ideas (a)–(b), we decided at the begin to implement approach (c).

We used algorithm (LA) that determines simultaneously R, Q and T_R, T_Q and which approximately realizes minimization $\min_{R,Q} \min_{T_R,T_Q}$ in (26). This is greedy method based on the entry list given by a permutation $\pi = (\pi(1), \ldots, \pi(o))$ of operations from the set O. For each resource (machine, form) we introduce the *release time*, i.e. time moment from which it becomes continuously available. Consecutive operation $\pi(i)$ from the list π is tested for possible allocation of available resources (machine, channel, forms) in time moments greater than release times of required resources. Such approach was used formerly in the DSS for RCPS, (Nowicki, Smutnicki, 1994). For each considered variant of allocation, the earliest possible completion time is calculated, taking into account head, body, tail and changeover times. The best, in this sense, allocation is accepted and chosen resources become occupied in proper periods of time, providing modification of resource release times. Constructive method

bases on the single run of LA for pre-defined π. There is a freedom in generating initial π – we propose to order operations according to nondecreasing d_j's. Improvement type algorithm we based on TS with the insert-type moves that change order of operations in π, being the entry to LA. Because of huge size of π, the important is the proper selection of perspective and non-perspective moves in the neighbourhood, to avoid too high computational cost of the single neighbourhood search, see (Nowicki, Smutnicki, 1998). This study has not been completed yet, however, one can hope that good properties associated with critical path and blocks defined for scheduling problems with makespan criterion, (Nowicki, Smutnicki, 1996, 1998, 2005b) can be implemented in this case as well.

The highest level of the algorithm in Figure 6 refers to batching strategy shown in Section 4. Note that we need to perform optimization using at most two decision variables (batch size, threshold). This can be done by an overview of pre-defined areas of parameter space.

7. COMPUTATIONAL EXPERIMENTS

Implementation. Algorithm has been implemented as a console application in C++ and run on a PC with 1GB RAM and processor with 1.86 GHz clock, (Budyn, 2007), and tested on the class of specially designed benchmarks.

Benchmarks. Computational experiments have been carried out by using benchmarks designed specially for coincidence with practical instances. Data for processing times are taken fundamentally from the production process.

Each benchmark is determined by four sets of data:

(1) models,
(2) freezer parameters,
(3) refrigerator parameters,
(4) order sizes.

Data for (2) and (3) are coded by: A – with equal processing times, B – with various processing times, whereas for (4) by: C – with small differences between size of orders, D – with large difference of sizes of orders. Combination of letters A and B with C and D is supplemented by the number of models, e.g. AD30 denotes 30 models with equal processing times and large variation between order sizes. The number of channels in each M^m and M^c is set to four. For instances containing 10 models experiments are carried out by using single machine of each type, for 20 models – using two, for 30 – using three, and for 40 – using four machines of each types. Orders are split into two batches.

Solution and schedule. Solution is represented by the permutation $\pi = (\pi(1), \ldots, \pi(o))$. Optimization criterion is the makespan. Detailed schedule is built from π by using the greedy method. To this order we start from the scratch schedule and add, in each step, single operation. Adding policy depends on machine type. Carousel machines perform doors for refrigerators and freezers and requires the couple of forms per model. Changeover times is comparable to operation processing

times but stops the whole carousel. For M^c, successive operation is allocated to that machine and channel which either is the earliest available under condition that owns required set of forms (changeover is not needed) or is the earliest available without any additional conditions (changeover is expected). To evaluate balance between these two these opposite tendencies we use priority index $\alpha\ C(a,c) + \beta\ C^*$ (the smallest value of the index is selected), where α, β are some parameters, and $C(a,c)$ is the release time of channel c on machine a and C^* is the earliest release time of machine that can process that operation without changeover. Based on the set of random solutions we set experimentally the best combination of parameters $\alpha = 0.25$ i $\beta = 0.75$.

Multichannel machines perform cupboards for refrigerators and requires 4 to 5 various forms per model. Three forms can be treated as allocated constantly to channel, next 1–2 are variable and depend on the model. Changeover times are relatively long. Operations are added using greedy scheme, similarly as the above. At first we check whether exists machine and channel having required set of forms – operation is allocated there. If there are no such configuration, algorithm looks for the earliest available machines having the missing forms – these forms are re-mounted on that channel which ensures the earliest completion of the operation, taking into account changeover time.

Algorithms. Three metaheuristic algorithms have been examined: genetic algorithm (GA), simulated annealing (SA), tabu search (TS). To ensure common conditions for comparisons, each algorithm uses the same representation of solution and schedule, as well as generates the same number of solutions equal 50*o, where o is the problem size. At the begin each algorithm has been analysed and tested separately to obtain the best configuration of components and tuning parameters. We skip consciously partial results obtained in this phase of algorithm development, and present only final results obtained through comparison the best selected versions of each method.

Genetic algorithm. There have been examined single and double point order crossover operators (SX, PMX, OX, OX-2), tournament and roulette wheel selection. GA has been tuned to set population size and mutation probability.

After extensive experiments there has been selected the best configuration: tournament selection, PMX crossover operator, mutation probability 0.05, population size 20.

Simulated annealing. There have been examined several cooling schemes, namely (a) automatic cooling (SA autotuning version, [1]), and a few manualy tuned cooling schemes: (b) geometric $T_k = \theta(a^k)$, (c) logarithmic $T_k = \theta(\ln^{-1}(k+1))$, (d) harmonic $T_k = \theta(k^{-1})$. For manually tuned schemes there have been also examined the influence of neighbourhood structure, initial temperature, cooling parameters as well as the number of repetitions in fixed temperature, on the quality of best solution found. Among all tested variants the one with autotuning and neighbourhood based on swap moves has turned out the champion.

Tabu search. There have been examined neighbourhoods based on adjacent pairwise interchange moves (deterministic) and based on the limited set of random swap moves (stochastic), two types of tabu list (complete permutations, recent move), tabu list length, diversification strategies. After extensive experiments there has been

set the best configuration: stochastic neighbourhood, tabu list based on recent moves of length 10, diversification after each 50 moves.

Evaluation methodology. Since algorithms are stochastic in their nature, each algorithm for each instance have been run 10 times. For each algorithm A and each run we evaluate the relative error (in per cent)

$$RE = 100\% \cdot \frac{C^A - C^{REF}}{C^{REF}} \tag{31}$$

where C^A is the makespan provided by algorithm A and C^{REF} is the reference makespan. As C^{REF} we use the best makespan found for this instance during all tests. Next, for each instance we calculate the BEST value of RE among 10 runs and Average value of RE among 10 runs.

Results. Results are shown in Tables 1 (makespan list) and 2 (relative error). Each run of the instance takes from a minute to few minutes on a PC. It is clear that the best results provides SA with autotunning, GA is worse, TS is the worst one. Conclusions from these results are numerous. First, more extensive further research should be made to develop GA as well as TS through application of special problem properties. Special attention need to be paid to promising TS approach, which can be improved by using generalisation of critical path and block notions.

Table 1. Makespans for algorithms GA, SA, TS. BEST* is the best one among all algorithms

		algorithm					
		GA		SA		TS	
benchmark	BEST*	BEST	AV	BEST	AV	BEST	AV
AC10	1056	1070	1097	1056	1086	1058	1093
AC20	1310	1351	1382	1310	1353	1310	1406
AC30	1240	1261	1340	1240	1291	1282	1376
AC40	1192	1228	1288	1192	1250	1278	1345
AD10	1101	1103	1138	1103	1136	1101	1142
AD20	1245	1321	1344	1245	1310	1314	1359
AD30	1219	1251	1296	1219	1312	1265	1430
AD40	1340	1400	1422	1340	1392	1388	1471
BC10	2307	2307	2357	2324	2380	2379	2402
BC20	4893	4969	5013	4893	4948	4938	5004
BC30	5379	5443	5469	5379	5440	5415	5486
BC40	4734	4734	4781	4758	4781	4758	4824
BD10	2844	2904	2919	2844	2913	2934	2952
BD20	4609	4677	4729	4609	4713	4669	4760
BD30	5397	5447	5494	5397	5467	5468	5532
BD40	5586	5592	5632	5586	5609	5628	5678

Table 2. Relative errors [%] to reference makespans (BEST*) for algorithms GA, SA, TS

	algorithm					
	GA		SA		TS	
benchmark	BEST	AV	BEST	AV	BEST	AV
AC10	1.3	3.9	0.0	2.8	0.2	3.5
AC20	3.1	5.5	0.0	3.3	0.0	7.3
AC30	1.7	8.1	0.0	4.1	3.4	11.0
AC40	3.0	8.1	0.0	4.9	7.2	12.8
AD10	0.2	3.4	0.2	3.2	0.0	3.7
AD20	6.1	8.0	0.0	5.2	5.5	9.2
AD30	2.6	6.3	0.0	7.6	3.8	17.3
AD40	4.5	6.1	0.0	3.9	3.6	9.8
BC10	0.0	2.2	0.7	3.2	3.1	4.1
BC20	1.6	2.5	0.0	1.1	0.9	2.3
BC30	1.2	1.7	0.0	1.1	0.7	2.0
BC40	0.0	1.0	0.5	1.0	0.5	1.9
BD10	2.1	2.6	0.0	2.4	3.2	3.8
BD20	1.5	2.6	0.0	2.3	1.3	3.3
BD30	0.9	1.8	0.0	1.3	1.3	2.5
BD40	0.1	0.8	0.0	0.4	0.8	1.6
AV all	1.87	4.02	0.09	2.99	2.21	6.01

8. EXTENSIONS AND CONCLUSIONS

In Figure 7 has been shown a small instance for operations "extrusion forming: X", where $X \in \{H, L\}$ on machines M^c, with relaxed changeover times, and equal batch size 9. Batches of the same order i has been denoted by $i.1$, $i.2$, etc. Obtained schedule follows from constructive version of LA method with π generated by FIFO rule (all d_j's are equal). This quite small instance clearly shows weakness of Gantt chart for huge production planning problems with complex time constraints – it is hardly to observe visually regularity on huge diagram, and to show possible direction of solution modification toward sub-optimal ones. This conclusion was formulated already in Nowicki, Smutnicki (1994) and is one of the reasons of stimulation of DSS development. Nevertheless, the programming package for Gantt chart presentation is the necessary tool for problem analysis.

As one can see, the study of the problem has not been completed yet. It is caused by the time – the problem is stated and applied very recently. The open subjects in particular areas leading to the solution algorithm, as well as alternative solution approaches, have been signalled in suitable sections. Valid is also the problem of storing semi-products before assembly line. It can be embedded in the presented model by introducing upper limit on waiting time of operations "extrusion forming: X", where $X \in \{H, L, B\}$ before final assembly. Advantageously, limits on waiting time can be transformed on appropriate arc weights in graph G.

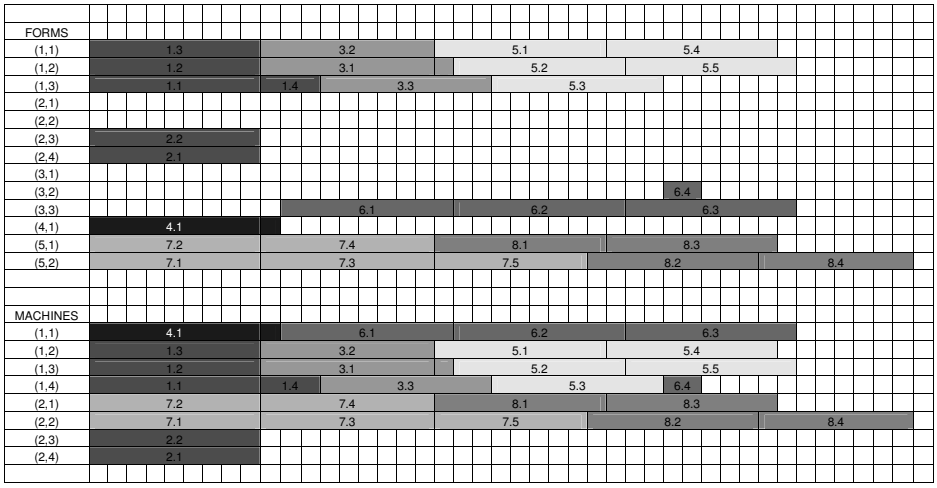

Fig. 7. Schedule instance

It is clear that the main challenge still remains optimization and synchronization of the work of the whole system. The proposed model is general enough to attack this problem as well.

REFERENCES

1. E.H.L. Aarts, P.J.M. van Larhoven, *Simulated annealing: a pedestrain review of the theory and some applications,* In: Patter Recognition and Applications, Eds. P.A. Devijver, J. Kittler, Springer, Berlin, 1987.

2. P. Brucker, A. Drexl, R. Möhring, K. Neumann, E. Pesch, *Resource-constrained project scheduling: Notation, classification, models, and methods,* European Journal of Operational Research 112 (1999), 3–41.

3. P. Brucker, S. Knust, A. Schoo, O. Thiele, *A branch and bound algorithm for the resource-constrained project scheduling problem,* European Journal of Operational Research 107 (1998), 272–288.

4. P. Budyn, *Scheduling under frequent changes of production profile,* MSc Thesis, ICECR, Wroclaw University of Technology, 2007.

5. F. Glover, M. Laguna *Tabu search,* Kluwer Academic Publishers, 1996.

6. W. Herroelen, E. Demeulemeester, B. De Reyck, *Resource-constrained project scheduling,* A survey of recent developments, Computers & Operations Research 25 (1998), 4, 279–302.

7. R. Kolisch, A. Drexl, *Local search for nonpreemptive multi-mode resource-constrained project scheduling,* IIE Transactions 29 (1997), 987–999.

8. R. Kolisch, S. Hartmann, *Heuristic Algorithms for Solving the Resource Constrained Project Scheduling Problem: Classification and Computational Analysis,* in: J. Węglarz (ed.), Handbook on Recent Advances in Project Scheduling, 147–178. Kluwer, 1999.

9. R. Kolisch, R. Padman, *An integrated survey of deterministic project scheduling,* Omega, 29 (2001), 249–272.

10. M. Laguna, R. Marti (2003) *Scatter search. Methodology and implementation in C,* Kluwer Academic Publishers, 2006.

11. M. Mori, C.C. Tseng, *A genetic algorithm for multi-mode resource constrained project scheduling,* European Journal of Operational Research 100 (1997), 134–141.

12. K. Neumann, C. Schwindt, N. Trautmann, *Advanced production scheduling for batch plants in process industries,* in: Günther H.O., Beek P, (eds.) Advanced Planning and Scheduling Solutions in Process Industry, Springer 2003.

13. E. Nowicki, C. Smutnicki, *A decision support system for resource constrained project scheduling problem,* European Journal of Operational Research 79 (1994), 183–195.

14. E. Nowicki, C. Smutnicki, *A fast taboo search algorithm for the job shop problem,* Management Science 6 (1996), 797–813.

15. E. Nowicki, C. Smutnicki, *Flow shop with parallel machines. A tabu search approach,* European Journal of Operational Research 106 (1998), 226–253.

16. E. Nowicki, C. Smutnicki a *Some new ideas in TS for job-shop scheduling,* in: Rego C. and Alidaee B. (eds.) Metaheuristic Optimization via Memory and Evolution: Tabu Search and Scatter Search, Kluwer Academic Publishers, 2005.

17. E. Nowicki, C. Smutnicki b, *New algorithm for the job-shop problem,* Journal of Scheduling 8 (2005), 145–159.

18. L. Özdamar, G. Ulusoy, An iterative local constrained based analysis for solving the resource constrained project scheduling problem. Journal of Operations Management 14 (1996), 3, 193–208.

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

20. A. Sprecher, A. Drexl, *Multi-mode resource-constrained project scheduling by a simple, general and powerful sequencing algorithm,* European Journal of Operational Research 107 (1998), 431–450.

21. P.R. Thomas, S. Salhi, *A Tabu Search Approach for the Resource Constrained Project Scheduling Problem,* Journal of Heuristics 4 (1998), 123–139.