Decision Making in Manufacturing and Services
Vol. 2 • 2008 • No. 1–2 • pp. 33–46

A Transfer Line Balancing Problem by Heuristic Methods: Industrial Case Studies

Olga Guschinskaya* Alexandre Dolgui**

Abstract. The paper deals with the problem of optimal configuration of a type of transfer lines which are equipped with transfer machines. Such machines perform operations with standard modular spindle heads which are activated sequentially. All operations assigned to the same spindle head (block of operations) are executed simultaneously by a set of tools fixed at the spindle head. The quantity of machines and spindle heads used to produce a part with the given productivity rate defines the final cost of the transfer line which must be minimized. To minimize this cost, a combinatorial problem of operations assignment to blocks and machines must be solved. The solution must provide a desired productivity (cycle time), it must also satisfy precedence and compatibility constraints. In this paper, we suggest improved versions of FSIC heuristic algorithm in order to help line designers to solve real-scale industrial problems. Results of computational experiments obtained for industrial cases are presented.

Keywords: Computer-aided design, machining, line balancing, optimization, heuristics

Mathematics Subject Classification: 90C27, 90B80, 90C59

Revised: 12 December 2008

1. INTRODUCTION

Machining transfer lines, dedicated to the mass production of a unique product or a family of similar products, are widely used in mechanical industry (Dashchenko, 2003). Designing such lines is a very complex problem due to necessity to take into account the manufacturing and design constraints at early design stage. At the same time, the manufacturers must provide their customers with the results of the preliminary design (a general line configuration and an estimation of its price) as quickly as possible. Moreover, the designers have to be able to respond interactively to all part modifications that the customer may provide them with even if the line design is already started. Therefore, the designers need the optimisation computed-aided

* Centre for Industrial Engineering and Computer Science, Ecole des Mines de Saint Etienne 158, Cours Fauriel, 42023 Saint Etienne Cedex 2, France. E-mail: guschinskaya@emse.fr
** Centre for Industrial Engineering and Computer Science, Ecole des Mines de Saint Etienne 158, Cours Fauriel, 42023 Saint Etienne Cedex 2, France. E-mail: dolgui@emse.fr

approaches to support the line design stage. Since transfer lines can be configurated differently, for example they can be equipped with machining centers with tools changers or with special transfer machines with multi-spindle heads, adequate mathematical models and efficient methods adapted to each concrete type of line are needed (Hitomi, 1996).

The considered in this paper transfer line configuration is the following: line consists of a number of linearly ordered multi-spindle machines that perform the operations required to manufacture parts. The advantages of multi-spindle machines are: high productivity, a reduced space and cost (Hitomi, 1996). There is at least one multi-spindle head at each machine. All the machines are linked by an automated material handling device without buffers between machines. The machining is done by tools fixed at the spindle heads. Each machine is equipped with several spindle heads. The activation of spindle heads at the same machine is sequential, due to the necessity to change the active spindle head or to move (reposition) the part. The reposition time can be assumed the same for all spindle heads, and equal to τ^b. The number of spindle heads at each station and the order of their activation must be determined during the line design process. Each spindle head incurs an investment cost and requires a certain amount of working space, therefore the total number of spindle heads which can be installed at the same machine is limited by number n_0.

Let $\mathbb{N}$ be the set of all operations needed for a part machining. Each operation $j \in \mathbb{N}$ is characterized by two parameters: the required working stroke length λ_j and the feed per minute s_j. The working stroke length includes the required depth of cut and the distance between the tool and the part surface. The feed is measured by how much the tool advances for each rotation of the tool. Each spindle head may have multiple tools to perform all the operations assigned to it simultaneously. Let N be a set of operations to be executed by the same spindle head. If set N contains only one operation, i.e. there will be only one tool fixed in the spindle head to execute this operation j, then the feed per minute $S(N)$ of the spindle head is equal to s_j and its working stroke length $L(N)$ is equal to λ_j. If set N contains several operations, then several tools are needed to perform them. In this case, the feed per minute of the spindle head is equal to $S(N) = \min\{s_j | j \in N\}$ and its working stroke length is equal to $L(N) = \max\{\lambda_j | j \in N\}$.

The objective of line design is to minimize the total number of spindle heads and machines (i.e. the line cost which is composed of fixed cost per machine and additional costs for spindle heads). This objective can be reached by an optimal assignment of the operations to spindle heads and to machines. The solution must provide a desired productivity as well as satisfy all technological constraints. To respect the cycle time objective, the total processing time of all operations executed at each machine must be less than the objective cycle time value, denoted by T_0. Loading and unloading of the part at all machines must be executed within the same cycle time, the loading and unloading time τ^s can be considered the same for all machines.

Taking into account the conceptual proximity to the well-known *Assembly Line Balancing Problem* (ALBP) (Baybars, 1986; Scholl and Klein, 1998), the described above problem was named the *Transfer Line Balancing Problem* (TLBP). In the wide range of different formulations of *Generalized Assembly Line Balancing Problem*

(GALBP) (Becker and Scholl, 2006) the most similar to TLBP is the known in the literature as the *Simple Assembly Line Design Problem* (SALDP) which deals with solving the assembly line balancing problem jointly with equipment selection (Bukchin and Tzur, 2000; Bukchin and Rubinovich, 2003; Gadidov and Wilhelm, 2000). Hence, for TLBP the operations must be grouped into blocks of parallel operations that define the machining parameters of the equipments. Thus, the set of alternative blocks is not known in advance and, as a consequence, TLBP cannot be reduced to SALDP as well as the methods used to solve SALDP cannot be applied directly for solving TLBP (Dolgui et al., 2006b).

In previous works, several exact methods were suggested for this novel problem (TLBP):

- an efficient method was suggested in (Dolgui et al., 2008), which transforms the initial problem into a constrained shortest path problem. Some dominance rules were developed to reduce the size of the obtained graph;
- a mixed integer programming (MIP) approach was developed in (Dolgui et al., 2006b).

Since TLBP is NP-hard, the exact algorithms are only applicable for small and medium-sized problems (less than 60 operations). For large-scale problems, *heuristic approaches* were proposed in the following papers:

- heuristic algorithms RAP and FSIC were suggested in (Dolgui et al., 2005). These heuristics are based on the COMSOAL method (Arcus, 1966), i.e. several operation assignments (in the increasing order of block and station indices) are randomly generated, and the best assignment is kept. RAP and FSIC differ mainly in operation selection for assignment to blocks and stations;
- an original technique of deterministic decomposition for MIP model was suggested in (Dolgui et al., 2006a);
- a heuristic multi-start decomposition approach was developed in (Guschinskaya et al., 2008).

In this paper, improved modifications for FSIC heuristic algorithm are suggested. The paper is organized as follows. Section 2 deals with the mathematical model of the studied problem. Section 3 presents solution methods. Section 4 is dedicated to experimental results. Concluding remarks are given in Section 5.

2. MATHEMATICAL MODEL

The following input data are assumed to be known:

- $\mathbb{N}$ is the set of all operations needed for a part machining;
- λ_j, $j \in \mathbb{N}$, the working stroke length required for operation j;
- s_j, $j \in \mathbb{N}$, the standard value of feed per minute for operation j;
- T_0 is the maximal admissible line cycle time (desired productivity);
- C_1 is the given cost of one machine;
- C_2 is the given cost of one spindle head;

- m_0 is the maximal authorized number of machines;
- n_0 is the maximal number of spindle heads (blocks) per machine;
- *precedence constraints* between the operations defining non-strict partial order relation over operation set $\mathbb{N}$. The precedence constraints can be represented by digraph $G = (\mathbb{N}, D)$. The arc $(i, j) \in \mathbb{N} \times \mathbb{N}$ belongs to set D if and only if operation j cannot precede operation i. Such precedence constraints are called non strict because if operation i is a predecessor of operation j, then either operation i can be executed before j or i and j can be executed simultaneously using a compound tool;
- *inclusion constraints* defining the groups of operations that must be assigned to the same machine, because of a required machining tolerance. These constraints can be represented by ES, a family of subsets from $\mathbb{N}$, such that all operations of the same subset $e \in ES$ must be assigned to the same machine. No two sets from ES can include the same operation; otherwise such sets should be aggregated;
- *machine exclusion constraints* defining the groups of operations that cannot be assigned to the same machine because of their technological incompatibility. These constraints can be represented by $\overline{ES}$, a family of subsets from $\mathbb{N}$, such that each subset $e \in \overline{ES}$ cannot be assigned to the same machine as a whole. Note that any proper subset of e can be assigned to the same machine, only the assignment of set e as a whole is forbidden;
- *block exclusion constraints* defining the groups of operations that cannot be assigned to the same spindle head because of their technological incompatibility. These constraints can be represented by $\overline{EB}$, a family of subsets from $\mathbb{N}$, such that the same subset $e \in \overline{EB}$ cannot belong to the same block as a whole. Note that any proper subset of e can be assigned to the same block, only the assignment of set e as a whole is forbidden.

Note: in order to obtain the strict precedence constraint between i and j (i strictly before j), it is necessary to include arc (i, j) to set D and the pair $\{i, j\}$ in $\overline{EB}$.

The block processing time $t^b(N_{kl})$ of operations N_{kl} belonging to the block l of machine k is determined as follows: $t^b(N_{kl}) = L(N_{kl})/S(N_{kl}) + \tau^b$. The spindle heads of the same machine are activated sequentially, therefore the machine processing time $t(N_k)$ is equal to the sum of its block processing times: $t(N_k) = \sum_{l=1}^{n_k} t^b(N_{kl}) + \tau^s$.

In order to reduce the number of feasible solutions a parameter m_0 is introduced which represents the maximal authorized number of machines. It can be obtained by analyzing the available plant area or the maximum authorized line investment cost, or calculated as an upper bound on m.

Thus, the considered transfer line balancing problem can be formulated as follows. Let P be a feasible solution for the considered problem. This solution can be represented by a collection $P = \{\{N_{11}, \ldots, N_{1,n_1}\}, \ldots, \{N_{m1}, \ldots, N_{m,n_m}\}\}$, determining an assignment of operations to a sequence of machines ($k = 1, 2, \ldots, m$) and repartition of operations assigned to the same machine k to n_k blocks. Let N_k be the work content (i.e. a set of operations) for machine k ($k = 1, 2, \ldots, m$) and N_{kl} be the set of operations grouped into common block l ($l = 1, 2, \ldots, n_k$) of machine k.

The objective function (1) estimates the line cost, i.e. the cost of solution P:

$$\textit{Minimize} \quad Q(P) = C_1 m + C_2 \sum_{k=1}^{m} n_k \tag{1}$$

Expression (2) is the cycle time constraint:

$$T(P) = \max\{t(N_k) | 1 \le k \le m\} \le T_0 \tag{2}$$

Constraints (3)–(4) reflect the fact that each operation from $\mathbb{N}$ must be included in a block once and only once:

$$\bigcup_{k=1}^{m} \bigcup_{l=1}^{n_k} N_{kl} = \mathbb{N}; \tag{3}$$

$$N_{k'l'} \cap N_{k"l"} = \emptyset, \quad (k'l') \neq (k"l"), \quad k', k" = 1, \ldots, m, \quad l' = 1, \ldots, n_{k'}, \quad l" = 1, \ldots, n_{k"} \tag{4}$$

Constraints (5) define the precedence relations on the set $\mathbb{N}$:

$$(k'-1)n_0 + l' \le (k''-1)n_0 + l'', \quad i \in N_{k'l'}, \quad j \in N_{k''l''}, \quad \forall (i,j) \in D \tag{5}$$

Constraints (6) determine the necessity of assigning certain operations to the same machine:

$$N_k \cap e \in \{\emptyset, e\}, \quad \forall e \in ES, \quad k = 1, \ldots, m \tag{6}$$

Constraints (7)–(8) deal with the impossibility of grouping certain operations into the same block or executing certain operations at the same machine, respectively:

$$e \not\subset N_{kl}, \quad \forall e \in \overline{EB}, \quad k = 1, \ldots, m, \quad l = 1, \ldots, n_k \tag{7}$$

$$e \not\subset N_k, \quad \forall e \in \overline{ES}, \quad k = 1, \ldots, m \tag{8}$$

Constraints (9) and (10) limit the number of machines and blocks, respectively:

$$m = m(P) \le m_0 \tag{9}$$

$$n_k = n_k(P) \le n_0, \quad k = 1, \ldots, m \tag{10}$$

In the next section, we present a heuristic algorithm for solving problem (1)–(10).

3. HEURISTIC METHODS

3.1. GENERAL SCHEME

The random search algorithm named FSIC (*First Satisfy Inclusion Constraints*) was suggested in (Dolgui et al., 2005). It deals with the special case of TLBP where all the operation times are given in advance (before grouping into blocks) and the block processing time is equal to the time of its longest operation. FSIC heuristic is

based on the COMSOAL method (Arcus, 1966), *i.e.* several operation assignments are randomly generated, and the best assignment is kept. In FSIC algorithm, all the operations related by inclusion constraints (and their non-assigned predecessors) are processed first.

Let list $L1$ be a list of operations that can be assigned to the current machine. Operation i can be assigned to current machine k if the following conditions hold:

- all its predecessors are already assigned;
- for all $e \in \overline{ES}$ if $i \in e$ then $e \cap (N_k \cup \{i\}) \neq e$;
- its assignment does not violate the cycle time constraint.

Let Op be the randomly chosen operation from list $L1$. If $Op \notin e$, $\forall e \in ES$, then algorithm tries to assign Op to the current block, if it is not possible (because of block exclusion constraints with the operations already assigned to the current block), it creates a new block at the current machine and tries to assign Op there if it is not possible (because of violating the cycle time constraint) it deletes this block, creates a new machine with one block and assign Op there. If impossible to create a new machine since the total number of machines will be greater than m_0 then the current solution is considered as not feasible and a new iteration of the algorithm starts.

If $Op \in ES$, then the algorithm uses an additional list $L2$. This list groups all the operations that must be assigned to the same machine as operation Op, i.e., the list is made of operation Op and all other operations from the corresponding set $e \in ES$ and all their non-assigned predecessors. The algorithm tries to assign all the operations from the list $L2$ to the current machine. If impossible, then the algorithm creates a new machine and tries to assign all the operations from the list $L2$ to the new machine. The algorithm stores the state of the current machine before the first assignment attempt of the list $L2$. It restores its state before the second try if the list $L2$ contains more than one operation. When all the operations from $L2$ have been assigned, $L1$ is rebuilt.

A complete iteration of this algorithm (described by Algorithm 1) gives a feasible solution for the transfer line (or the conclusion that the solution of this iteration is not feasible). The number of iterations executed by the algorithm during available computational time is denoted TR_{tot}. The algorithm returns the best assignment found through TR_{tot} iterations. The value of parameter *seed* is changed to have different random solutions at each iteration.

In this paper, a number of modifications for this heuristic is suggested. It is adapted for the case where operations are given by their parameters (feed per minute and working stroke) and not by their processing times. The general scheme of the method is represented by Algorithm 1. The following notations are used: P_{min} is the best solution,P_{cur} is the current solution, C_{min} is the cost of the best solution, C_{cur} is the cost of the current solution, TR_{tot} is the current number of iterations, TR_{nimp} is the number of iterations without improving C_{min}, TR_{tot_aut} and TR_{nimp_aut} are the maximal allowed values of TR_{tot} and TR_{nimp}, T_{cur} is the current solution time, T_{res} is the available solution time.

Algorithm 1

Step 0: Set $C_{min} = \propto$, $TR_{tot} = 0$, $TR_{nimp} = 0$, $seed = 0$.

Step 1: Set $m = 1$, $C_{cur} = C_1$,

while $T_{cur} < T_{res}$ **and** $TR_{nimp} < TR_{nimp\_aut}$ **and** $TR_{tot} < TR_{tot\_aut}$.

Step 2: Build list $L1$ of operations that can be assigned
to the current machine k as follows:
A non assigned operation i is added to $L1$,
if all following conditions are satisfied:
1) **for** all $e \in \overline{ES}$ **if** $i \in e$ **then** $e \cap (N_k \cup \{i\}) \neq e$;
2) **either** $\exists l \in \{1, \ldots, n_k\}$ such that $t(N_k) \leq T_0$ for $N_{kl} = N_{kl} \cup \{i\}$,
or $t(N_k) + \lambda_j / s_j + \tau^b \leq T_0$ **and** $n_k + 1 \leq n_0$;
3) all predecessors of operation i are assigned, i.e.
if $(j, i) \in D$ **then** $j \in \bigcup_{r=1}^{k} \bigcup_{l=1}^{n_r} N_{rl}$.

Step 3: **If** $L1 \neq \emptyset$,
then choose an operation i randomly from list $L1$, set list $L2 = \{i\}$,
else go to *Step 8*.

Step 4: **If** $\exists e \in ES$ such that $i \in e$,
then add to list $L2$ all operations from e
and all their non-assigned predecessors.

Step 5: Try to assign each operation from list $L2$ to current machine k
verifying all the constraints.
If it is possible, **then** go to *Step* 7.

Step 6: Set $C_{cur} = C_{cur} + C_1 + C_2 n_k, \quad k = k + 1$.
If $k > m_0$, **then** set $C = \propto$ and go to *Step* 9,
else try to assign each operation from list $L2$ to new machine k
verifying all the constraints.
If it is not possible, **then** set $C = \propto$ and go to *Step* 9,

Step 7: **If** all the operations of **N** have been assigned, **then** go to *Step* 9.
else go to *Step* 2.

Step 8: Set $C_{cur} = C_{cur} + C_1 + C_2 n_k$, $k = k + 1$.
If $k > m_0$, **then** set $C = \propto$ and go to *Step* 9, **else** go to *Step* 2.

Step 9: **If** $C_{min} > C_{cur}$ **then** set $C_{min} = C_{cur}$, $TR_{nimp} = 0$, $P_{min} = P_{cur}$,
else set $TR_{nimp} = TR_{nimp} + 1$.

Step 10: Set $TR_{tot} = TR_{tot} + 1$.

end while

Five modifications of this basic version of FSIC heuristic are proposed in this paper. In order to distinguish the different combinations of these modifications, the control parameters are introduced that define if the corresponding modification is applied or not.

3.2. MODIFICATION 1 – ASSIGNMENT OF OPERATION OP

In the basic version of FSIC when Op is chosen randomly from list $L1$ the algorithm tries to assign it to the current block if impossible then a new block is created. Two modifications are suggested:

1. At first try to assign Op to all previous blocks of the current machine, then to the current block.
2. At first try to assign Op to all previous blocks of all existing machines in their creation order, then to the current block.

The control parameter `check_blocks` can take the following values: 0 for the basic version, 1 for checking all blocks of the current machine and 2 for checking all previous blocks.

3.3. MODIFICATION 2 – ASSIGNMENT OF THE OPERATIONS FROM LIST L2

In the basic version, if it is impossible to assign all the operations from list $L2$ to the current machine then a new machine is opened, if it is impossible to assign all operations from list $L2$ to this new machine, this machine is deleted and another operation Op is randomly chosen from list $L1$. The following modification is suggested: if it is impossible to assign all operations from list $L2$ to the current machine, then another operation Op is randomly chosen from list $L1$ without opening a new machine. This modification avoids creating new machines, if there remain operations that can be assigned to the current machine. The control parameter `2_trials_L2` can take the following values: 1 for the basic version, 0 for the modified one.

3.4. MODIFICATION 3 – GENERATING OF LIST L1

In the basic version of FSIC heuristic an operation can be added to list $L1$ if its assignment to the current machine does not lead to the violation of the cycle time constraint. Hence, if the control parameter `check_blocks` $= 2$ and if the cycle time constraint is checked for the current machine then an operation, that can be assigned to a previous block not belonging to the current machine, cannot be included in $L1$. Moreover, in practice the time checking can take long time (because of calculating the block times based on the operations parameters) and often none operation is discarded, especially when the current machine has only one opened block. On other hand, including an operation which assignment can violate the cycle time constraints leads to the creation of a new machine if this operation was randomly chosen. In order to check which variant outperforms another one, the control parameter `check_time` is introduced, it can take the following values: 0 for checking the time of operations when the list $L1$ is building and 1 for no checking.

3.5. MODIFICATION 4 – ORGANISATION OF LIST L2

List $L2$ includes the operations related by an inclusion constraint and their non-assigned predecessors. In the basic version the operations to be assigned are chosen randomly from list $L2$. The following modification is suggested: divide list $L2$ into two parts: the first one with non-assigned predecessors and second one with the operations related by the inclusion constraint. At first operations to be assigned are chosen randomly from the first part and when it is empty the operations from the second part are assigned. The control parameter `divide_L2` can take the following values: 0 for the basic version and 1 for the case when the operations belonging to list $L2$ are divided into two parts.

4. EXPERIMENTAL STUDY

4.1. GENERATION OF TESTS BASED ON INDUSTRIAL PROBLEMS CHARACTERISTICS

The purpose of this study is to compare the performances of FSIC heuristic having different modifications (see Section 3) while solving the real industrial problems, to do it a set of tests based on the properties of industrial problems was generated. Usually, the machining process includes milling, drilling, boring, etc. a set of part elements such as planes, facets, holes of different types (cylindrical, bevel, threaded, etc). Each element concerns a certain side (or surface) of the part and is characterized by a set of required machining operations. Usually the designers describe the part to be machined using typical machining features. The concept of feature associates the technological characteristics of a machining element with its geometric parameters. There are different types of features and they differ one from another by the number and nature of machining operations that they include. An example of a part containing 5 features is represented in Figure 1. To create the tests with characteristics close to real life problems the most common features used in industrial application were generated.

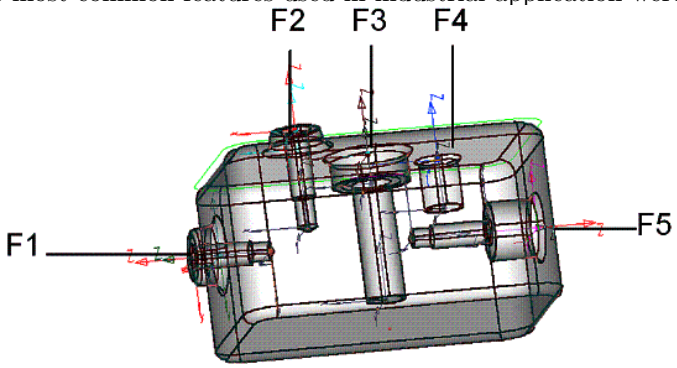

Fig. 1. Part to be machined

The dataset was generated including 4 data series of 30 test instances each. The series with different problem sizes were generated in order to study the problems of

different complexity. The data series contain the problems with 10, 20, 30 and 40 features, respectively. Within each series the test instances have the same number of features but the type of each feature and the side of part where it had been placed were chosen randomly in order to simulate different part configurations. Since different feature types contain different number of machining operations the number of operations varies from instance to instance in the same data series. The minimal, average and maximal numbers of operations are reported for each series in Table 1.

Four modifications concerning the techniques used by heuristic are suggested. Each method is defined by the values of 4 control parameters, `check_blocks` $\in \{0, 1, 2\}$, `2_trials_L2`, `check_time` and `divide_L2` $\in \{0, 1\}$. Thus, each method can be labelled by a number having 4 positions each contains the value of the corresponding parameter. For example, method 2010 has `check_blocks` $= 2$, `2_trials_L2` $= 0$, `check_time` $= 1$, and `divide_L2` $= 0$. All methods were implemented using C++, the experiments were carried out on HP workstation XW 6200 (3,6 GHz, 2 Gb RAM). In order to establish the best techniques to use each series was solved by $24 = 3{\cdot}2{\cdot}2{\cdot}2$ variants of FSIC heuristic with constant parameters $TR_{tot} = 100\,000$, $TR_{nimp} = 100\,000$ and for different resolution times T_{res}. For the presentation of the obtained results the following notations are used: Δ_{max}, Δ_{av}, Δ_{min} – maximal, average and minimal deviation of the line cost obtained by a method from the best value obtained, respectively; *NBS* – number of obtained best solutions ($\Delta = 0$); *PBS* – percent of *NBS* to the total number of tests.

It can be observed that if `check_blocks` $= 2$ then the methods having `2_trials_L2` $= 0$ always outperform the methods having `2_trials_L2` $= 1$. For the methods having `check_blocks` $= 0$ or 1 the situation is opposite: the methods with `2_trials_L2` $= 1$ are better than with `2_trials_L2` $= 0$. It also can be noticed that whilst the number of features augments the methods having `check_blocks` $= 2$ improve their performances in relation to other methods. It can be mentioned that for each series min *NBS* for the shorter times is inferior to min *NBS* for the longer times, the same for max *NBS* and only for the 40 features series they are equal.

In Table 2 the result obtained for all 4 series are summarized, the following notations are used: *NBA* the number of obtained best solutions for all instances ($\Delta = 0$); *PBA* – percent of *NBA* to the total number of tests (240 including solving test with short and long times); $NB1$ the number of obtained best solutions while solving instances with short available time; $PB1$ – percent of $NB1$ to the total number of tests (120); $NB2$ the number of obtained best solutions while solving instances with long available time; $PB2$ – percent of $NB2$ to the total number of tests (120).

The obtained results show that for the all solution times the best methods are 2011, 2010 and 2001. Thus, it can be concluded that the best techniques are to check all previous blocks while assigning an operation and do not create a new machine if the operations from list $L2$ are not assigned to the current machine. It can be mentioned that while solving the problems with small times another methods found

Table 1. Results

a) Results for the 1^{st} series: $n_f = 10$, $|N_{min}| = 29$, $|N_{av}| = 38$, $|N_{max}| = 47$.

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	2000	2001	2010	2011	2100	2102	2110	2111
$T_{res} = 1.5$																								
Δ_{max}	16.1	16.1	12.8	12.8	8.7	8.7	7. 9	7.9	16.1	16.1	16.1	12.9	9.7	9.7	6. 5	6.5	8.7	8.7	5.3	5.3	5.7	5.7	6.4	6.5
Δ_{av}	5.5	5.4	4.8	4.5	1.6	1.5	0.9	1.1	5.2	5.3	4.8	4.4	1.8	1.9	1.0	0.8	1.3	1.0	1.0	0.9	1.1	1.0	1.1	1.4
NBS	7	6	8	10	18	18	24	23	8	8	8	7	17	18	21	23	19	21	22	22	21	21	20	19
T_{res} =15																								
Δ_{max}	13.3	13.3	10.0	10.0	8.7	8.7	5.0	5.0	13.3	13.3	13.3	10.6	10.0	10.0	5.0	5.0	8.7	8.7	3.9	3.9	6.7	6.7	6.7	13.3
Δ_{av}	3.64	3.53	2.45	2.73	0.77	0.84	0.25	0.47	3.68	3.61	3.53	3.05	1.17	1.39	0.59	0.59	0.8	0.74	0.46	0.46	1.18	1.03	0.7	3.64
NBS	12	13	15	14	24	23	28	26	12	12	11	13	22	22	25	25	23	24	25	25	22	23	25	12

b) Results for the 2^{nd} series: $n_f = 20$, $|N_{min}| = 46$, $|N_{av}| = 55$, $|N_{max}| = 92$.

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	2000	2001	2010	2011	2100	2102	2110	2111
$T_{res} = 3$																								
Δ_{max}	37.0	35.2	31.5	35.2	13.0	10.9	11.4	11.4	36.4	37.0	29.6	29.6	9.1	13.0	9.2	9.1	6.8	6.8	4.6	4.6	11.4	11.4	9.3	9.1
Δ_{av}	11.8	11.6	10.9	10.8	4.0	3.7	3.6	3.7	11.8	12.0	10.3	10.6	4.3	4.7	4.2	4.0	1.6	1.4	0.7	0.8	2.8	2.9	3.2	3.0
NBS	6	6	6	6	8	9	8	9	2	2	2	2	5	3	3	3	13	14	21	19	7	10	6	6
$T_{res} = 30$																								
Δ_{max}	34.0	34.0	30.2	32.1	11.3	7.1	8.3	7.1	34.0	30.2	32.1	30.2	12.5	10.7	8.1	8.9	5.7	5.7	5.4	3.9	5.7	5.9	5.9	5.9
Δ_{av}	14.5	14.1	12.9	12.8	4.8	3.7	3.2	3.3	13.1	13.2	12.6	12.0	4.5	4.3	4.0	4.0	1.1	1.1	1.1	1.0	2.3	2.2	2.2	2.1
NBS	7	7	7	7	7	10	9	10	7	7	7	8	10	10	9	9	19	20	19	17	10	12	13	13

Table 1. (continued)

c) Results for the 3^{rd} series: $n_f = 30$, $|N_{min}| = 80$, $|N_{av}| = 84$, $|N_{max}| = 127$.

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	2000	2001	2010	2011	2100	2102	2110	2111
$T_{res} = 4.5$																								
Δ_{min}	7.0	7.0	7.5	7.5	1.2	1.2	2.3	2.3	7.0	7.0	5.8	5.8	4.3	2.8	2.3	2.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Δ_{max}	44.6	40.5	39.8	36.9	27.5	22.7	20.2	20.2	36.9	36.1	37.3	32.5	15.5	19.3	15.5	15.6	10.1	6.0	5.5	4.9	6.0	5.5	5.5	4.9
Δ_{av}	23.6	22.8	22.5	21.8	11.8	11.1	9.8	9.4	21.0	21.2	20.0	19.8	9.6	9.3	8.2	8.2	2.1	1.8	1.2	0.9	2.1	2.0	2.0	2.5
NBS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	12	12	16	3	4	4	2
T_{res} =45																								
Δ_{max}	93.3	93.3	93.3	93.3	21.5	21.5	20.0	20.3	93.3	93.3	93.3	93.3	15.0	17.5	14.1	16.3	5.1	7.0	5.1	5.4	5.1	6.4	7.7	6.4
Δ_{av}	28.0	27.5	26.2	25.3	11.3	10.8	10.3	9.6	26.0	26.2	24.9	24.7	9.7	9.8	8.7	9.1	1.5	1.6	1.5	1.2	2.2	2.6	2.5	2.2
NBS	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	14	13	14	16	11	9	8	10

d) Results for the 4^{th} series: $n_f = 40$, $|N_{min}| = 115$, $|N_{av}| = 141$, $|N_{max}| = 158$.

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	2000	2001	2010	2011	2100	2102	2110	2111
$T_{res} = 6$																								
Δ_{min}	19	18.3	15.7	16.2	9.87	9.87	10.5	10.5	17	17.7	17.7	17.7	8.2	8.2	10.5	9.15	0	0	0	0	0	0	0	0
Δ_{max}	61	60	51.5	53.9	36.5	31.3	33	28.7	53	53	54	54	26.1	25	22.9	25.2	7.46	7.92	8.47	8.21	5.38	5.67	8.51	6.96
Δ_{av}	35.1	35.6	34.3	33.7	20.6	20.1	19.3	18.5	31.1	31	30.5	29.8	16.5	16	15.4	15.3	3.22	3.3	2.42	2.34	2.59	2.25	3	2.24
NBS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	6	8	8	6	5	2	6
T_{res} =60																								
Δ_{min}	17.2	16.6	16.2	16.2	10.6	9.93	6.62	10.6	14.6	14.6	15.2	16.2	9.23	8.46	9.23	9.23	0	0	0	0	0	0	0	0
Δ_{max}	53.5	53.5	51.5	53.1	35.4	25.3	30.1	27.4	48.5	48.5	43.4	43.4	20.2	20.2	21.6	20.2	7.34	6.32	6.33	5.26	5.71	4.59	4.63	5.1
Δ_{av}	32.1	31.9	32.4	30.9	17.8	17.3	16.8	16.3	29.9	29.9	28.9	29	15	14.9	14.8	14.1	2.39	2.08	2.09	1.15	2.12	2.29	2.01	2.27
NBS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	6	7	13	1	2	6	5

in average 24 best solutions for 120 problems that is 19.56 %, and for solving the problems with long times they found in average 38 best solutions for 120 problems that is 31.35%. That confirms that the choice of efficient method is more important for the shorter resolution time. The best method 2011 found 65 (54.2%) best solutions and 71 best solutions (59.2%) while solving the tests with short and long available time, respectively.

Table 2. Total results for 4 series

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011
NBA	38	38	42	43	63	66	75	74	35	35	34	36
PBA	15.8	15.8	17.5	17.9	26.2	27.5	31.2	30.8	14.5	14.5	14.1	15.0
*NB*1	13	12	14	16	26	27	32	32	10	10	10	9
*PB*1	10.8	10	11.6	13.3	21.7	22.5	26.7	26.7	8.33	8.33	8.33	7.5
*NB*2	25	26	28	27	37	39	43	42	25	25	24	27
*PB*2	20.8	21.7	23.3	22.5	30.8	32.5	35.8	35.0	20.8	20.8	20.0	22.5

	1100	1101	1110	1111	2000	**2001**	**2010**	**2011**	2100	2102	2110	2111
NBA	60	59	64	66	107	**116**	**128**	**136**	81	87	84	86
PBA	25.0	24.5	26.6	27.5	44.5	**48.3**	**53.3**	**56.6**	33.7	36.2	35.0	35.8
*NB*1	22	21	24	26	46	**53**	**63**	**65**	37	41	32	33
*PB*1	18.3	17.5	20	21.7	38.3	**44.2**	**52.5**	**54.2**	30.8	34.2	26.7	27.5
*NB*2	38	38	40	40	61	**63**	**65**	**71**	44	46	52	53
*PB*2	31.7	31.7	33.3	33.3	50.8	**52.5**	**54.2**	**59.2**	36.7	38.3	43.3	44.2

5. CONCLUSION

An important industrial problem was considered: the configuration of transfer lines equipped by multi-spindle machines. These machining systems are known to be costly and in order to be efficient, they must be well-designed. The design of these lines is a complex combinatorial problem, known to be NP-hard. As a consequence, it often requires a great computation time for designers. Therefore, using heuristic methods is indispensable in order to obtain solutions of good quality for real-scale industrial problems in acceptable time. In this paper, four improved versions of FSIC heuristic have been suggested. These methods have been tested on 4 data series, each of which contained 30 problems generated with taking into account the real machining parts configurations. The obtained numerical results have demonstrated improving the method performances. Further investigations will concern implementing heuristic approaches with learning and backtracking.

Acknowledgments

This research was financially supported by INTAS under Project 03-51-5501.

REFERENCES

Arcus A.L., 1966: COMSOAL: A Computer Method of Sequencing Operations for Assembly Lines. *International Journal of Production Research*, 4, 259–277.

Baybars I., 1986: A Survey of Exact Algorithms for the Simple Assembly Line Balancing. *Management Science*, 32, 909–932.

Becker C., Scholl A., 2006: A Survey on Problems and Methods in Generalized Assembly Line Balancing. *European Journal of Operational Research*, 168, 694–715.

Bukchin J., Tzur, M., 2000: Design of Flexible Assembly Line to Minimize Equipment Cost. *IIE Transactions*, 32, 585–598.

Bukchin J., Rubinovich J., 2003: A Weighted Approach for Assembly Line Design with Station Paralleling and Equipment Selection. *IIE Transactions*, 35, 73–85.

Dashchenko A., I. (Ed), 2003: *Manufacturing Technologies for Machines of the Future 21st Century Technologies*. Springer.

Dolgui A., Finel B., Guschinskaya O., Guschinsky N., Levin G., Vernadat F., 2006a: Balancing Large-scale Machining Lines with Multi-spindle Heads Using Decomposition. *International Journal of Production Research*, 44 (18-19), 4105–4120.

Dolgui A., Finel B., Guschinsky N., Levin G., Vernadat F., 2006b: MIP Approach to Balancing Transfer Lines with Blocks of Parallel Operations. *IIE Transactions*, 38, 869–882.

Dolgui, A. Finel, B., Guschinsky, N., Levin, G., and Vernadat, F., 2005. A Heuristic Approach for Transfer Lines Balancing. *Journal of Intelligent Manufacturing*, 16(2), 159–171.

Dolgui A., Guschinsky N., Levin G., Proth J.-M., 2008: Optimisation of Multi-position Machines and Transfer Lines. *European Journal of Operational Research*, 185(3), 1375–1389.

Gadidov R., Wilhelm W., 2000: A Cutting Plane Approach for the Single-product Assembly System Design Problem. *International Journal of Production Research*, 38(8), 1731–1754.

Guschinskaya O., Dolgui A., Guschinsky N., Levin G., 2008: A Heuristic Multi-start Decomposition Approach for Optimal Design of Serial Machining Lines. *European Journal of Operational Research*, 189(3), 902–1013.

Hitomi K., 1996: *Manufacturing Systems Engineering*. Taylor & Francis.

Scholl A., Klein R., 1998: Balancing Assembly Lines Effectively: a Computational Comparison. *European Journal of Operational Research*, 114, 51–60.