A MILP approach for robust transfer line design

Aleksandr Pirogov, Evgeny Gurevsky, André Rossi, Alexandre Dolgui

A MILP approach for robust transfer line design

(presented at the 7^{th} IESM Conference, October 11–13, 2017, Saarbrücken, Germany)

Aleksandr Pirogov*, Evgeny Gurevsky†, André Rossi‡ and Alexandre Dolgui*
* LS2N, IMT Atlantique, Nantes, France
† LS2N, Université de Nantes, France
‡ LERIA, Université d'Angers, France

***Abstract*—We address the design stage of transfer lines subject to precedence constraints, fixed cycle time, limited number of machines and uncertain tasks. The objective is to build the line configuration, so as to maximize its robustness, i.e., its ability to withstand task time variability. To do this, we formulate a mixed-integer linear program (MILP), where the concept of stability radius is used as robustness measure and calculated in two classic metrics, l_1 and l_∞. To handle these two MILP models, the commercial solver CPLEX 12.6.2 is used. The corresponding computational results are reported and discussed at the end of this paper.**

***Keywords*—transfer lines, robust optimization, stability radius**

I. Introduction

Transfer lines are automated flow-oriented manufacturing systems, dedicated to yield a large quantity of product and usually designed for a period of 7-10 years (7 days a week and 24 hours a day) of functioning. The machines of such a line are organized in a linear manner, connected by a conveyor belt without intermediate buffers. Each of them is equipped by one or several blocks, where production tasks are executed in parallel using multi-spindle heads. We consider the lines where the blocks of any machine are activated sequentially in a predefined order, one by one without significant interruptions. As concerns production parts, they start their execution, in a raw state, on the first machine and pass through all of them in the order of their location so as to be turned into a finished state. The machines work simultaneously, performing their own and same set of tasks on successive parts. The machine completes its execution on the current part when all its blocks have been activated and have accomplished their operation. Since the considered lines are paced, all parts situated on the conveyor belt are transferred simultaneously to the next corresponding machines, when the functioning of these latter are completed. The time period between two successive transfers is called the *cycle time*, which is equal to the working time of the most loaded machine and determines, among others, the moment for one outgoing and new one incoming parts.

As regards the production tasks, they can not be executed in an arbitrary manner, but subject to *precedence constraints*, usually described by a directed acyclic graph, where the set of vertexes (resp. arcs) represents the set of tasks (resp. a partial order of tasks execution). Thus, for example, the arc (i, j) means that the task j can not be performed before the task i, but i and j can be executed simultaneously inside a block.

The design of such a line can be usually viewed as an optimization problem, which, with respect to a given production goal and subject to known industrial constraints, consists in assigning all production tasks to blocks and allocate these latter to machines, determining inside them the order of blocks activation. The first works, appeared in international refereed journals and dealing with the representation of the design stage for transfer lines as optimization problem, were published by Dolgui et al. in [4]–[6], where the authors consider a cost-oriented objective function aiming to minimize a weighted sum of the numbers of used machines and blocks, respectively. These pioneer journal articles have finally motivated many other studies on this problem, among which we can cite the works of Battaïa et al. on reductions approaches in [2], [3], Guschinskaya et al. on meta-heuristic methods in [8], [9], and Dolgui and Ihnatsenka on a branch-and-bound algorithm in [7].

For all mentioned above papers, the task processing time was considered to be deterministic and the number of available machines was supposed to be unlimited, but in reality this is not necessarily truthful. Thus, for example, production space restrictions often impose the limits on the number of used machines. As for task processing times, they can be modified from their nominal values because of numerous delays and micro-stoppages during manufacturing or because of the changes of product specification required by customer. In this work, we show how to integrate these two important industrial aspects into the design stage of transfer lines through an original robust approach, which has a significant advantage for its application with respect to classic *stochastic* [1] or *fuzzy* [10] methods, consisting in the fact that there is no need to possess reliable historical data on the variability of task processing times. Namely, we model our purpose as optimization problem within the framework of mixed-integer linear programming (MILP), seeking a transfer line configuration with fixed number of machines, which is able to support, without violating its admissibility, the greatest amplitude of task time variability. The numerical value of this amplitude is customary called in the literature as the *stability radius* and usually calculated for two classic metrics, l_1 and l_∞. As in our previous paper [11], dealing with assembly lines, in this article, we continue to demonstrate that the stability radius can be also used as a useful and veritable measure of solution robustness, contrary to the popular works on scheduling problems with uncertain jobs, published in [12], where it was studied as an indicator of sensitivity for already known optimal solutions.

The rest of the paper is organized as follows. Section II describes the main notations of the studied optimization problem and formalizes the notion of stability radius. Section III introduces the necessary decision variables and presents a MILP model for each metric of stability radius computing. Computational results, provided by the commercial solver

CPLEX 12.6.2, are given in Section IV. Conclusion and perspectives are delivered in Section V.

II. Problem statement

Below, we introduce some useful notations, which are necessary to the further formalization of the studied optimization problem.

- $V = \{1, 2, \ldots, n\}$ is the set of necessary tasks;
- $\widetilde{V} \subseteq V$ is the set of all uncertain tasks, whose processing time can be modified during manufacturing;
- $W = \{1, 2, \ldots, m\}$ is the set of available machines;
- t_j is the nominal processing time of the task $j \in V$;
- ξ_j is a non-negative variation of the processing time of the uncertain task $j \in \widetilde{V}$;
- ρ_1 (resp. ρ_∞) is the value of stability radius, computed in the metric l_1 (resp. l_∞);
- T is the cycle time;
- $r_{\max}$ is the maximal number of tasks, which can be assigned to block;
- $b_{\max}$ is the maximal number of blocks, which can be allocated into one machine;
- $G = (V, A)$ is a directed acyclic graph representing the precedence constraints, where A is the set of arcs.

The value of $b_{\max}$ is not given *a priori*, but it can not be greater than $\max\left\{k \mid \sum_{i=1}^{k} t_{\pi_i} \leq T\right\}$, where π_i is the task index of the permutation of V with respect to the non-decreasing order of their processing times. The latter expression corresponds to the case of tasks allocation to machine, without violating the cycle time and requiring the greatest number of blocks. In what follows, we will use this upper bound as the value of $b_{\max}$.

The knowledge of $b_{\max}$ permits us to simplify the representation of line configuration as shown in Figure 1, where

- $U = \{1, 2, \ldots, mb_{\max}\}$ is the set of indices of all potential blocks;
- $U(p) = \{(p-1) \cdot b_{\max} + 1, \ldots, p \cdot b_{\max}\}$ is the set of indices of all blocks potentially belonging to the machine p, $p \in W$.

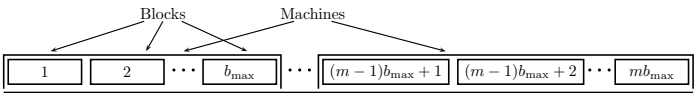

Fig. 1: Enumeration of the set of blocks

Based on the introduced notations, the studied problem can be formally viewed as the assignment of all tasks to blocks on the pre-defined line configuration, shown in Figure 1, subject to the cycle time and precedence constraints, which maximizes the value of ρ_1 (resp. ρ_∞) and preserves the assignment admissibility, whatever the total amplitude of uncertain tasks variations $\sum_{j \in \widetilde{V}} \xi_j$ (resp. $\max_{j \in \widetilde{V}} \xi_j$) not exceeding ρ_1 (resp. ρ_∞).

Let us consider an illustrative example, shown in Figure 2, where an allocation of the set of tasks $\{1, 2, \ldots, 6\}$ is given. For the presented example, the cycle time is equal to 8.5, indicated by a vertical dotted line, there are no precedence constraints and only one machine having two blocks is used. The first block contains the tasks 1, 2 and 5, while the second block is made of the tasks 2, 4 and 6. Each block is contoured by a dashed line. The tasks are represented by rectangles, whose length determines the corresponding processing time. The uncertain tasks $\{1, 4, 5\}$ are colored in gray. It is not difficult to see that for any total variation $\xi_1 + \xi_4 + \xi_5 \leq 2$, the allocation remains admissible, but a slightly greater value, where, for example, $\xi_1 = 2 + \delta$, $\xi_4 = 0$, $\xi_5 = 0$, violates its admissibility. This means that the stability radius in the l_1-metric equals 2. Regarding the l_∞-metric, it is easy to see that the considered allocation supports any variation ξ_1, ξ_4, and ξ_5 not exceeding 1.5, however it is not especially true for little bigger values, where, for example, $\xi_1 = 1.5 + \delta$, $\xi_4 = 1.5 + \delta$, $\xi_5 = 0$. The latter, following the definition of stability radius, indicates that $\rho_\infty = 1.5$.

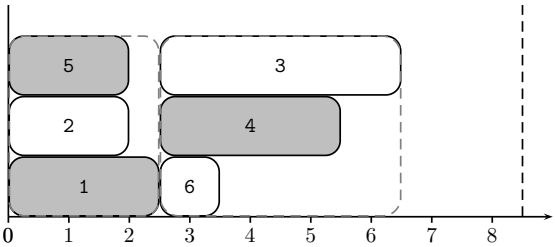

Fig. 2: Example of tasks allocation

III. MILP formulations

In this section, the MILP models for two metrics l_1 and l_∞, noted respectively by P_1 and P_∞, are described.

P_1 is formulated on the following decision variables:

- $\rho_1 \geq 0$ is the stability radius value to maximize;
- $x_{j,k}$ is a binary variable that is set to one if and only if the task j is allocated to the block k;
- y_k is equal to 1 if the block k is not empty and 0, otherwise;
- $\tau_k \geq 0$ determines the time of the block k;
- $\Delta_p \geq 0$ represents the minimal value of *save time* among all the blocks allocated into the machine p;
- a_p is a non-negative variable that is positive if the machine p has at least one assigned uncertain task.

The notions of block time and block save time are explained in Figure 3, where the block save time corresponds to the difference between block time and the longest processing time among the uncertain tasks assigned to this block. Thus, in Figure 3, the save time is equal to 1.5 for this first block and 0 for the second one.

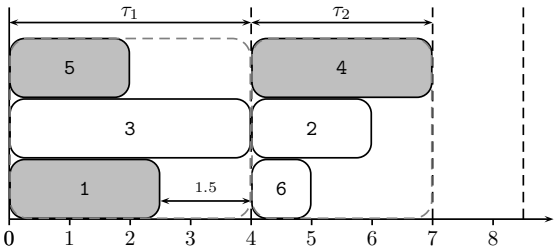

Fig. 3: Illustration of block time and block save time

An important result, simplifying the modeling of P_1, can be expressed by the following theorem.

Theorem 1. *The stability radius ρ_1 is calculated as follows*

$$\rho_1 = \min_{p \in \widetilde{W}} \left(T - \sum_{k \in U(p)} \tau_k + \Delta_p \right), \tag{1}$$

where $\widetilde{W}$ is a set of machines having at least one uncertain task.

P_∞ is formulated on the variables $x_{j,k}$, y_k and τ_k with the same sense as for P_1 and the following additional ones:

- $\rho_\infty \geq 0$ is the stability radius value to maximize;
- $\xi_{j,k}$ is the variation of the task j on the block k.

A. MILP formulation for P_1

Objective function:

$$\text{Maximize } \rho_1$$

Each task has to be allocated to exactly one block:

$$\sum_{k \in U} x_{j,k} = 1, \quad \forall j \in V \tag{2}$$

No more than $r_{\max}$ tasks can be included to block :

$$\sum_{j \in V} x_{j,k} \leq r_{\max}, \quad \forall k \in U \tag{3}$$

The block having at least one assigned task is considered as not empty, otherwise empty:

$$x_{j,k} \leq y_k, \quad \forall k \in U, \ \forall j \in V \tag{4}$$

$$y_k \leq \sum_{j \in V} x_{j,k}, \quad \forall k \in U \tag{5}$$

Any block has to be empty, if the previous block on the same machine is empty:

$$y_{k+1} \leq y_k, \quad \forall p \in W, \ \forall k \in U(p) \setminus \{pb_{\max}\} \tag{6}$$

The block time is the maximal processing time over the tasks allocated to this block:

$$t_j \cdot x_{j,k} \leq \tau_k, \quad \forall k \in U, \ \forall j \in V \tag{7}$$

The definition of the minimal save time for each machine:

$$\Delta_p \leq T \cdot (1 - y_k) + \tau_k - t_j \cdot x_{j,k}, \quad \forall p \in W, \ \forall k \in U(p), \ \forall j \in \widetilde{V} \tag{8}$$

The load of any machine can not be greater than the cycle time:

$$\sum_{k \in U(p)} \tau_k \leq T, \quad \forall p \in W \tag{9}$$

Precedence constraints:

$$\sum_{q=k}^{|U|} x_{i,q} \leq \sum_{q=k}^{|U|} x_{j,q}, \quad \forall (i,j) \in A, \ \forall k \in U \setminus \{1\} \tag{10}$$

Constraint (11) sets a_p according to its definition, and a_p is used in constraint (12) to compute ρ_1 by Theorem 1:

$$x_{j,k} \leq a_p, \quad \forall p \in W, \ \forall k \in U(p), \ \forall j \in \widetilde{V} \tag{11}$$

$$\rho_1 \leq T \cdot (2 - a_p) - \sum_{k \in U(p)} \tau_k + \Delta_p, \ \forall p \in W \tag{12}$$

To reduce the search space, it is possible to improve this model using the so-called assignment intervals:

$$x_{j,k} = 0, \quad \forall j \in V, \ \forall k \notin Q(j) \tag{13}$$

where $Q(j) = [l_j, u_j]$ is the interval of indexes of the blocks able to accommodate the task j, $j \in V$. Here,

$$l_j = \left\lceil \frac{|P(j) \cup \{j\}|}{r_{\max}} \right\rceil,$$

$$u_j = mb_{\max} + 1 - \left\lceil \frac{|S(j) \cup \{j\}|}{r_{\max}} \right\rceil,$$

$P(j)$ and $S(j)$ are the sets of predecessors and successors of the task j in the precedence graph, respectively.

B. MILP formulation for P_∞

The main idea of the MILP formulation for P_∞ is focused on the fact that the processing time of all uncertain tasks can be increased by ρ_∞ without loosing the feasibility for the optimal solution.

Constraints (2) – (6), (9) and (10) are the same for models P_∞ and P_1. The difference is expressed by constraints (14) – (17).

Objective function:

$$\text{Maximize } \rho_\infty$$

The block time is determined separately for blocks with uncertain tasks and without them:

$$t_j \cdot x_{j,k} + \xi_{j,k} \leq \tau_k, \quad \forall k \in U, \ \forall j \in \widetilde{V} \tag{14}$$

$$t_j \cdot x_{j,k} \leq \tau_k, \quad \forall k \in U, \ \forall j \in V \setminus \widetilde{V} \tag{15}$$

Variation of any uncertain task can not exceed the cycle time value, whatever its allocation:

$$\xi_{j,k} \leq T \cdot x_{j,k}, \quad \forall k \in U, \ \forall j \in \widetilde{V} \tag{16}$$

As mentioned above, the processing time of any uncertain task can be increased by the value of ρ_∞:

$$\rho_\infty = \sum_{k \in U} \xi_{j,k}, \quad \forall j \in \widetilde{V} \tag{17}$$

IV. Computational results

A set of 12 randomly generated benchmark instances has been used to examine the MILP model of P_1 and P_∞, where we distinguish 5 small and 7 medium size of them. Their main parameters are given in Table I, whose columns comprise the instance number $\#$, its number of tasks n, number of fixed machines m, the cycle time T and the order strength of the precedence constraints, $OS = \frac{2|A|}{n(n-1)}$, represented by the graph $G = (V, A)$ and expressed in percents.

TABLE I: Benchmark instances

$\#$	n	m	T	$OS, \%$
Small size				
1	8	4	25.5	75.00
2	11	6	10.5	58.18
3	9	5	9	83.33
4	11	4	67.5	60.00
5	7	4	9	52.38
Medium size				
6	29	11	37.5	50.74
7	28	8	162	22.49
8	21	7	19.5	70.95
9	32	9	2100	83.47
10	25	8	19.5	71.67
11	30	11	37.5	44.83
12	35	10	60	59.90

The computational results were carried out on a MacBook Pro having Intel Core i5 2.7 GHz and 8 GB RAM. The commercial solver CPLEX 12.6.2 was used through the Concert Technology C++ modeling layer for addressing the MILP models.

For each benchmark instance, six tests were generated by varying the number of uncertain tasks $|\widetilde{V}| \in \{\lceil \frac{n}{4} \rceil, \lceil \frac{n}{2} \rceil\}$ and the maximal number of tasks per block $r_{\max} \in \{1, 2, 3\}$. The results for each series are reported respectively in Tables II–VII, where for each metric, three columns indicate respectively the lower and upper bounds on the optimal value of stability radius, provided by CPLEX, as well as the corresponding CPU time (sec.), which is less than 600 sec., if an optimal solution was found before reaching this test time limit. In the case, where CPLEX was not able to find either lower or upper bound, the abbreviation "**n/a**" is put in the corresponding cell of the Table.

Analyzing the obtained results, we can remark that an optimal solution was found for all series of small size benchmark instances for P_1 and P_∞ within the limits of 2.51 sec. As concerns the medium size benchmark instances, the situation is different. Thus, for P_1, an optimal solution was found only for the $8th$ instance of each series, except the last one. For P_∞, we can note some improvements, but the necessary time for finding optimal solutions remains quite high. Another interesting observation is the greater the $r_{\max}$ value, the easier P_∞ to solve. The summary of found feasible and optimal solutions is given in Table VIII.

TABLE II: $|\widetilde{V}| = \lceil \frac{n}{4} \rceil$ and $r_{\max} = 1$

$\#$	LB_1	UB_1	CPU	LB_∞	UB_∞	CPU
1	4.5	4.5	0.02	4.5	4.5	0.04
2	1.5	1.5	0.29	1.5	1.5	0.31
3	1	1	0.03	1	1	0.03
4	41.5	41.5	1.07	21.5	21.5	1.04
5	0	0	0.03	0	0	0.01
6	1.5	74.6	600	6.75	12.5	600
7	40	409.9	600	33	55	600
8	6.5	6.5	175.7	4.25	4.25	321.57
9	148	4091.6	600	400	700	600
10	4.5	13	600	3.25	6.5	600
11	2.5	87.9	600	5.75	18.5	600
12	**n/a**	**n/a**	600	**n/a**	**n/a**	600

TABLE III: $|\widetilde{V}| = \lceil \frac{n}{4} \rceil$ and $r_{\max} = 2$

$\#$	LB_1	UB_1	CPU	LB_∞	UB_∞	CPU
1	13.5	13.5	0.09	13.5	13.5	0.02
2	4.5	4.5	1.24	4.5	4.5	0.13
3	3	3	0.05	3	3	0.04
4	55.5	55.5	1.61	55.5	55.5	0.3
5	4	4	0.06	4	4	0.01
6	12.5	74.6	600	12.5	12.5	55.99
7	55	444.8	600	55	92	600
8	10.5	10.5	243.97	10.5	10.5	10.01
9	454	4200	600	700	700	315.56
10	6.5	13	600	6.5	6.5	7.84
11	8.5	102.3	600	18.5	18.5	34.65
12	**n/a**	**n/a**	600	3	60	600

TABLE IV: $|\widetilde{V}| = \lceil \frac{n}{4} \rceil$ and $r_{\max} = 3$

$\#$	LB_1	UB_1	CPU	LB_∞	UB_∞	CPU
1	13.5	13.5	0.11	13.5	13.5	0.02
2	4.5	4.5	0.66	4.5	4.5	0.14
3	3	3	0.06	3	3	0.03
4	55.5	55.5	1.75	55.5	55.5	0.2
5	4	4	0.1	4	4	0.02
6	12.5	75	600	12.5	12.5	43.45
7	55	324	600	55	92	600
8	10.5	10.5	213.28	10.5	10.5	2.67
9	454	5719.1	600	700	700	98.87
10	6.5	37.3	600	6.5	6.5	10.42
11	14.5	75	600	18.5	18.5	14.62
12	**n/a**	**n/a**	600	1	174.4	600

TABLE V: $|\widetilde{V}| = \lceil \frac{n}{2} \rceil$ and $r_{\max} = 1$

$\#$	LB_1	UB_1	CPU	LB_∞	UB_∞	CPU
1	4.5	4.5	0.02	2.25	2.25	0.02
2	1.5	1.5	0.37	1.25	1.25	0.57
3	1	1	0.03	0.5	0.5	0.03
4	23.5	23.5	1.11	12.75	12.75	2.01
5	0	0	0.04	0	0	0.03
6	3.5	75	600	3.17	12.5	600
7	**n/a**	**n/a**	600	9.75	55	600
8	3.5	3.5	453.08	1.75	6.2	600
9	84	4599.7	600	80	700	600
10	1.5	12.5	600	3.17	12.5	600
11	3.5	74.3	600	3.75	12.5	600
12	**n/a**	**n/a**	600	**n/a**	**n/a**	600

V. Conclusion

This paper deals with finding a transfer line configuration having the greatest stability radius subject to various industrial constraints. The stability radius is evaluated in both l_1 and l_∞

TABLE VI: $|\widetilde{V}| = \lceil \frac{n}{2} \rceil$ and $r_{\max} = 2$

#	LB_1	UB_1	CPU	LB_∞	UB_∞	CPU
1	13.5	13.5	0.06	13.5	13.5	0.04
2	4.5	4.5	1.65	4.5	4.5	0.14
3	3	3	0.07	3	3	0.05
4	29.5	29.5	1.93	29.5	29.5	0.78
5	3	3	0.05	3	3	0.03
6	3.5	75	600	12.5	12.5	335.45
7	47	429.8	600	49.5	55	600
8	6.5	6.5	167.47	6.5	6.5	22.22
9	242	5369.6	600	700	700	250.02
10	6.5	13	600	6.5	6.5	9.46
11	6.5	75	600	12.5	18.5	140.05
12	**n/a**	**n/a**	600	**n/a**	**n/a**	600

TABLE VII: $|\widetilde{V}| = \lceil \frac{n}{2} \rceil$ and $r_{\max} = 3$

#	LB_1	UB_1	CPU	LB_∞	UB_∞	CPU
1	13.5	13.5	0.1	13.5	13.5	0.02
2	4.5	4.5	1.76	4.5	4.5	0.31
3	3	3	0.07	3	3	0.05
4	29.5	29.5	2.51	29.5	29.5	0.32
5	3	3	0.04	3	3	0.02
6	12.5	75	600	12.5	12.5	117.42
7	55	324	600	55	55	190.39
8	6.5	7.5	600	6.5	6.5	18.52
9	700	4200	600	700	700	89.29
10	6.5	36	600	6.5	6.5	21.27
11	5.5	102.6	600	12.5	12.5	39.15
12	1	174.2	600	4	20	600

TABLE VIII: Number of found solutions

$\lvert\widetilde{V}\rvert$	$\lceil \frac{n}{4} \rceil$			$\lceil \frac{n}{2} \rceil$		
$r_{\max}$	1	2	3	1	2	3
Feasible solutions for P_1	11	11	11	10	11	12
Optimal solutions for P_1	6	6	6	6	6	5
Feasible solutions for P_∞	11	12	12	11	11	12
Optimal solutions for P_∞	6	11	11	5	10	11

metrics. For each metric, the corresponding MILP model was elaborated. Numerical results show that the used commercial solver can find an optimal solution for small size instances in less than 3 sec., but unfortunately not able to provide satisfactory outcomes for medium size ones.

The proposed MILP models are a first attempt for applying robust optimization approaches for transfer line design. The second natural step of our future research is a detailed analysis of the studied problem in order to find its new structural and combinatorial properties permitting to develop more efficient resolution methods, based on appropriate reduction rules.

ACKNOWLEDGEMENT

This work was partially supported by the council of the french region “Pays de la Loire”.

REFERENCES

[1] K. Ağpak and H. Gökçen, *A chance-constrained approach to stochastic line balancing problem*, European Journal of Operational Research, 180(3): 1098–1115, 2007.

[2] O. Battaïa and A. Dolgui, *Reduction approaches for a generalized line balancing problem*, Computers & Operations Research, 39(10): 2337–2345, 2012.

[3] O. Battaïa, E. Gurevsky, F. Makssoud and A. Dolgui, *Equipment location in machining transfer lines with multi-spindle heads*, Journal of Mathematical Modelling and Algorithms in Operations Research, 12(2): 117–133, 2013.

[4] A. Dolgui, B. Finel, F. Vernadat, N. Guschinsky, G. Levin, *A heuristic approach for transfer lines balancing*, Journal of Intelligent Manufacturing, 16(2): 159–172, 2005.

[5] A. Dolgui, B. Finel, N. N. Guschinsky, G. M. Levin, F. B. Vernadat, *MIP approach to balancing transfer lines with blocks of parallel operations*, IIE Transactions, 38(10): 869–882, 2006.

[6] A. Dolgui, N. Guschinsky, G. Levin, J.-M. Proth. *Optimisation of multi-position machines and transfer lines*, European Journal of Operational Research, 185(3): 1375–1389, 2008.

[7] A. Dolgui and I. Ihnatsenka, *Branch and bound algorithm for a transfer line design problem: Stations with sequentially activated multi-spindle heads*, European Journal of Operational Research, 197(3): 1119–1132, 2009.

[8] O. Guschinskaya, A. Dolgui, N. Guschinsky and G. Levin, *A heuristic multi-start decomposition approach for optimal design of serial machining lines*, European Journal of Operational Research, 189(3): 902–913, 2008.

[9] O. Guschinskaya, E. Gurevsky, A. Dolgui and A. Eremeev, *Metaheuristic approaches for the design of machining lines*, International Journal of Advanced Manufacturing Technology, 55(1-4): 11–22, 2011.

[10] Y. Kara, T. Paksoy and C.-T. Chang, *Binary fuzzy goal programming approach to single model straight and U-shaped assembly line balancing*, European Journal of Operational Research, 195(2): 335–347, 2009.

[11] A. Rossi, E. Gurevsky, O. Battaïa and A. Dolgui, *Maximizing the robustness for simple assembly lines with fixed cycle time and limited number of workstations*, Discrete Applied Mathematics, 208: 123–136, 2016.

[12] Y. Sotskov, N. Sotskova, T.-C. Lai and F. Werner, *Scheduling under Uncertainty: Theory and Algorithms*, Belorusskaya nauka, Minsk, 2010.