Vol.2 pp. 7-13 **Journal of Operational Management & Marketing Strategies (OMMS)**

Minimization of a Transfer Line Cycle Time

Sana Bouajaja, Najoua Dridi

OASIS Laboratory, National Engineering School of Tunis (ENIT)

University of Tunis Elmanar, BP. 37 le Belvédère, 1002, Tunis, Tunisia

sana_bouajaja@yahoo.fr, najoua.dridi@enit.rnu.tn

Abstract— In this paper, we deal with the optimal reconfiguration of a transfer line. Such a line is composed of serial stations. Operations of the same station are partitioned into blocks activated sequentially. The operations of each block are executed simultaneously by the same spindle-heads. The problem is to group the operations into blocks and to assign them to machines in order to minimize the line cycle time in such a way all operations are assigned and constraints are satisfied. A heuristic approach is proposed to solve the problem. For the solution enhancement, improvement procedures are suggested. A lower bound on the line cycle time is defined to measure the efficiency of the proposed algorithm. Test instances are performed and experimental results are presented.

Keywords— Reconfigurable transfer line, Assignment and balancing problems, Optimization, heuristics.

I. INTRODUCTION

A Transfer Line Balancing Problem (TLBP) is studied here. It consists on the optimization of serial machining lines. These lines are reconfigured in industry for mass production [1], [2], where high production rate is required. In such lines, parts to be produced are simultaneously transferred from a station to the next at the end of a line cycle, as shown in Fig. 1. The line cycle time is the maximum of station times. Each station is equipped with spindle head having several tools to execute several operations at the same time. Multi-spindle heads are also called blocks. The objective is to assign all operations to blocks then to allocate the obtained blocks to stations, minimizing the cycle time. This is similar to line balancing problem.

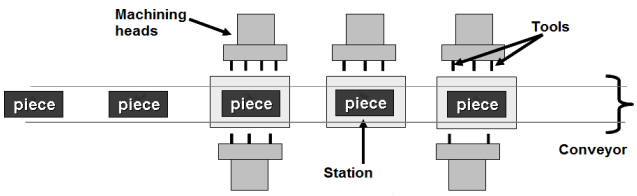

Fig. 1. A scheme of a machining line

II. STATE OF THE ART

The balancing problem arises for all types of production lines. Originally, it was formulated for mono-product assembly lines, in the automotive industry. The objective of the problem is to assign operations to workstations, while the precedence constraints are satisfied and each operator has the time to realize all operations that are assigned to him during the cycle time [3]. In the literature, this problem is known as the Simple Assembly Line Balancing Problem (SALBP).

In 1986, several models have been listed in [4]: ***SALBP:*** only the most common constraints are considered: the cycle time and precedence constraints. Depending on the objectives, several versions of SALBP are considered, we cite only: *SALBP-1*(minimizing the number of stations with a given cycle time) and *SALBP-2* (minimizing the cycle time with a given number of stations).

The SALBP problem doesn't reflect the industrial reality, because of the considered simplifying assumptions. Therefore, more general assumptions have been introduced; in this case we speak of Generalized Assembly Line Balancing Problem (GALBP).

GALBP: This model considers both current constraints and other less common as: grouping operations (must be assigned to the same station), incompatibility (operations must be assigned to different stations), parallel stations (operations performed simultaneously)...

Our problem called Transfer Line Balancing Problem (TLBP) can be considered as mentioned in [5] by Dolgui and Guischinskaya, as a version of GALBP. To our knowledge, studies dealing with this problem aren't numerous. A series of studies was carried out by Dolgui and his team. Several families of problems have been addressed in [4], assuming that: the activation mode of blocks is sequential, parallel or mixed, the set of possible blocks is available in advance or not.

For solving these problems, exact methods have been proposed. They are based on:
- ***Graph theory***: in [7], the original problem has been transformed into a shortest path problem, after the construction of a special graph to the problem.
- ***The mixed integer linear programming***: the TLBP problem was presented in [8] as a mixed performance integer linear program. In [10], the authors have improved the of their approach, reducing the number of variables by a more detailed analysis of the original problem constraints and the calculation of a lower bound for the station number.
- ***Branch and Bound****:* This method has been proposed in [9]. In this paper, a lower bound for the objective function has been developed; it is based on a relaxation of the problem studied, in order to transform it to a "Partitioning Problem Set" and to calculate a lower bound for the station number. As Transfer Line Balancing Problems are NP-hard [9],

ISSN: 2961-662X

heuristic methods have been used for solving large instances. Some heuristics have been proposed in [7]:
- ***RAB algorithm (Random Assignment of Blocks):*** this heuristic based on COMSOAL (Computer Method of Sequencing Operations for Assembly Lines) technique, can build stations progressively, by a random assignment of possible blocks to the current station.
- ***DFS algorithm (Depth-First Search):*** heuristic based on the technique of depth-first search to find the shortest path in a special graph. The algorithm stops when the first solution is found.
- ***Mixed optimization approach:*** presented in [5], it is based on the decomposition of a heuristic solution on sub-problems which are resolved by an exact method to improve the quality of the initial solution.

In the majority of studies found in the literature, considering the TLBP problem, the optimization criterion is to the line cost by reducing the number of stations and blocks. While in this work, we aim to minimize the cycle time of the line.

III. PROBLEM DESCRIPTIONS

The problem studied in this paper consists on the assignment of all the operations necessary to produce the final product. This problem is introduced by analogy to SALBP-2, where the criterion is to minimize the cycle time.

A. Constraints

Different constraints are considered in this problem:

1) *Precedence constraints:* the execution order of the operations may be partially specified due to the technological constraints, they can be illustrated by a graph that contains nodes corresponding to the operations and arcs connecting the nodes. The arc (i, j) exists if the operation i can be performed before or simultaneously with the operation j.
2) *Exclusion constraints:* express the impossibility of combining some operations in the same block or in the same station.
3) *Inclusion constraints:* express the fact that two operations must be executed in the same block or in the same station.

In addition, limitations are imposed on stations and blocks. In fact, the capacity of stations in terms of machining units (n_0) must be taken into consideration. Indeed, the maximum emplacement to equip the stations is limited. In addition, a machining unit has a limited number of tools (i_0) that perform several operations at the same time.

B. Parameters and Notations

The parameters used are:

N: the set of all operations,

m_0: the fixed number of stations in the line,

n_0: the maximum number of blocks per station,

i_0: the maximum number of operations per block,

t_i: the execution time of operation i, $i \in N$.

E^s (E^b): subsets of operations so that all operations of each subset must be assigned to the same station (block)

$\underline{E^s}$ ($\underline{E^b}$): subsets of operations so that all operations of each subset can't be assigned to the same station (block)

Problem constraints can be represented as follows:

$G^r(N,D^r)$: a directed graph representing the precedence constraints between operations.
$G^s=(N,E^s)$ (respectively $G^b=(N,E^b)$): a graph representing the inclusion constraints for operations in the same station (respectively block). For $X \subseteq N$, $X \in E^s$ (respectively E^b) if and only if the operations of X must be assigned to the same station (respectively block).
$\overline{G}^s = (N, \underline{E}^s)$ (Respectively $\overline{G}^b = (N, \underline{E}^b)$): a graph modeling exclusion constraints for operations in the same station (respectively block). For $X \subseteq N$, $X \in \underline{E}^s$ (respectively $\underline{E}^b$) if and only if the operations of X can't be assigned to the same station (respectively block).

C. Objective

To solve the optimization problem we have to determine the reconfiguration parameters: $N_k = \{N_{k1},..., N_{kn_k}\}$ the set of blocks in station k (k=1…m_0), where N_{kl} is the set of operations grouped into the same block l (l = 1…n_k) of station k, $P = \langle N_1,...,N_{m_0} \rangle = \{\{N_{11},...N_{1n_1}\},...,\{N_{k1},...,N_{kn_k}\},...,\{N_{m_1},...,N_{mn_{m_0}}\}\}$:a reconfiguration decision presenting an assignment of operations to a series of machines (k=1…m_0) and repartition of operations to n_k blocks of the same station k, and minimizing the cycle time of the line. Fig. 2 illustrates the line structure.

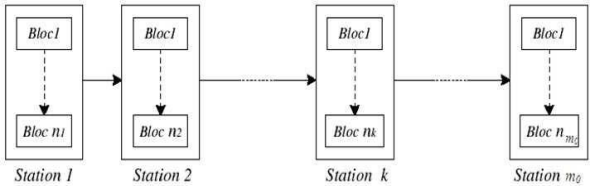

Fig. 2. A Machining Line structure

The cycle time of the line T_c is calculated as follows: The activation mode of blocks that governs how to engage the machining units of the same station is sequential, so the working time of a station is equal to the total execution time of its units:

$$T_k = \sum_{l=1}^{n_k} T(N_{kl}), \forall k = 1,..., m_0 \quad (1)$$

The block running time $T(N_{kl})$ from the station k depends on the operations set N_{kl}. Operations in the same block are executed in parallel, so the execution time of a block is the maximum time of its operations.

ISSN: 2961-662X

$$T(N_{kl}) = Max \left\{ t_j / j \in N_{kl} \right\} \quad (2)$$

As the line cycle time T_c is the time to process a product by any station (time of the bottleneck station), we have:

$$T_c = \underset{k=1\ldots m_0}{Max} \left(\sum_{l=1}^{n_k} Max \left\{ t_j / j \in N_{kl} \right\} \right) \quad (3)$$

Since the TLBP is a generalization of the simple assembly line problem known to be NP-hard, the considered problem is also a complex problem and can't be resolved by exact methods.

IV. Resolution Approach

In this section, we describe the overall approach we developed to solve our problem. This approach proceeds in four steps:

A. The Precedence graph transformation algorithm

In [8], a precedence graph transformation algorithm has been proposed to reduce the problem size. Indeed, the inclusion constraints to the same block can be treated in advance. Using G^r and E^b, the N set can be divided into subsets called macro-operations grouping the operations that must be performed in the same block. This transformation reduces the problem size and eliminates the E^b constraints. For more details refer to [8].

Our work takes place after this step and it is performed with the macro-operations, obtained after this transformation. In the following, we use the term operation to refer to macro-operation.

B. Heuristic method

As the problem is NP-hard, we propose a heuristic to achieve a compromise between computation time and quality of the obtained solution. The developed heuristic method provides an initial solution called T_{c_heur}, i.e. all operations are assigned to stations and blocks respecting all the problem constraints.

The steps of this heuristic are:

4) Assigning rank to the vertices of G^r

Thereby we determine the assignment order of operations without violating the precedence constraints. If L_r denotes the operations list of the same rank r, we proceed to the assignment of an operation from L_r if all operations of the set L_x ($1 \le x \le r$) are affected, and therefore all its predecessors are already affected. Thus, operations are assigned to the adequate station and the appropriate block, in ascending order of their rank.

5) Allocation to the station S_k ($1 \le k \le m_0$)

The assignment of the current operation i to the station S_k must satisfies the following constraints: Inclusion station, precedence, and exclusion station.

6) Allocation to the block b_{kl} ($1 \le l \le n_0$)

Once the current operation i is affected to the station S_k, we define the block b_{kl} of S_k where i will be performed. This assignment must satisfy the following constraints: Inclusion block, precedence, exclusion block and the capacity of the block i_0.

C. Improvement procedures

7) Improvement by exchange between blocks

In order to improve the heuristic solution i.e. to assure further reduction of the obtained line cycle time T_{c_heur} which depends on the bottleneck station S_g, we proceed to move some operations of this station from one block to another in the same station.

We move from the block b_{gl} to an adjacent block, the operation $i_{max\ gl}$ which imposes the cycle time of this block, if all the constraints are respected.

We opted for an exchange between adjacent blocks to reduce the risk of violating the precedence constraints. The idea of this procedure is to start with the movements that help to bring the best improvement, i.e to reduce the cycle time of the bottleneck station T_{cg}. After each movement, the cycle time is calculated and the new bottleneck station is identified to apply the same procedure again. If no exchange produced a decrease in cycle time, the procedure stops.

8) Improvement by exchange between stations

Reducing the cycle time of the bottleneck station can also be obtained by moving some operations from this station to the adjacent stations. So that the precedence relations are not violated, we move from the last block of S_g the operation having the greatest operating time $i_{\max gn_g}$ to the first block of S_{g+1} and move $i_{max\ gl}$ from the block b_{g1} to the last block of S_{g-1}. The direction of travel is shown schematically in Fig 4.

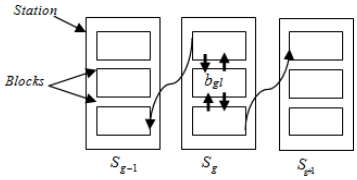

Fig. 3. Direction of operations travel between blocks and stations

When one of the problem constraints is not satisfied moving to the new station becomes impossible. It should be noted that any movement must ensure the reduction of the line cycle time. These movements can cause the appearance of a new bottleneck from the two affected stations S_{g-1} and S_{g+1}.

We must therefore ensure that, after these changes, the new cycle time (which corresponds to execution time of the new bottleneck station) is less than the old value of T_c, if not these changes are discarded. To estimate the quality of the heuristic solution and the improved solution or to give proof of their optimality in some special cases, we have developed a lower bound for T_c.

V. Lower Bound

An obvious lower bound for the cycle time of the line T_c is given:

ISSN: 2961-662X

$$BI1 = \underset{i=1...N}{Max}(t_i) \quad (4)$$

This bound corresponds to the case where there is a single block per station and is generally so far from the optimal value of the criterion considered.

We propose another lower bound *BI*2, better than *BI*1, because it takes into account some constraints (exclusion blocks, limitation of the blocks number per station n_0 and limitation of the operations number per block i_0).

The calculation of this bound is obtained by relaxing the precedence constraints, so we form blocks with operations in the order of decreasing durations and place in each block the maximum number of operations.

Thus, operations of large execution time occupy the same block, and will be performed simultaneously; hence the cycle time will be reduced. In order to achieve our goal, we have also interest to occupy all stations. To minimize the cycle time, the distribution of blocks on the stations must seek to balance the load of different stations.

The steps of *BI*2 calculation are:

A. Construction of blocks

Take the N operations in descending order of their operating time, and form blocks with the maximum number of operations i_0.

We obtain n blocks $b_1, b_2,..., b_n$, ($n = \lfloor \frac{N}{i_0} \rfloor + 1$. $\lfloor a \rfloor$ is the integer part of a) with durations such that:

$T(b_1) > T(b_2) > ... > T(b_n)$ and the execution time of a block b_k:

$$T(b_k) = \underset{i \in b_k}{Max}(t_i), \forall k = 1...n.$$

During the construction of these blocks, if the current operation i to assign to the block under construction don't respect the exclusion block constraints, it's left to a later stage and we move to the next operation and verify the same thing.

B. Distribution of blocks to stations

First, we assign the m_0 first blocks b_k at each station. The execution time of each station S_k, at this stage, is $T(S_k) = T(b_k)$. If there are blocks not yet allocated, the current block is placed in the station with the lowest execution time. Repeat until all blocks are assigned.

At each block's assignment to a station, we have to update the station execution time. At the end, we should have balanced stations in terms of execution time thus the cycle time is minimized.

VI. Experimental results

To evaluate the performance of the proposed method and to identify the impact of some parameters (density $D(G^r)$ of the precedence graph G^r and the problem size i.e the number of operations N), an experimental study is developed. We used the Microsoft Visual Studio C++ version 6.0 under Windows XP and the programming language C++, on a PC Intel ® Pentium ® with 1.73 GHz frequency and 512 MB of RAM.

16 series of tests were randomly generated for a number of operations N varying from 5 to 30 and a graph density $D(G^r)$ ranging from 0.11 to 0.7 . Each series includes 10 instances of the same N and $D(G^r)$ but differ by the operations time.

For each of the 160 instances, we apply the heuristic algorithm to find an initial solution called S_{heur} with a line cycle time T_{c_heur}. To improve S_{heur} i.e to reduce T_{c_heur} the two improvement procedures were used. The final solution is $S_{exchange\_s}$ and the cycle time is noted T_{c_impr}.

A. Performance indicators

Four performance indicators are used to measure the quality of the resolution method:

- *Time_execution (s):* is the computational time related to the heuristic *method and improvement algorithm.*
- *Improvement (%):* is the improvement percentage of S_{heur}, calculated as follows:

$$improvement(\%) = \frac{(T_{c\_heur} - T_{c\_impr})}{T_{c\_heur}} \times 100 \quad (5)$$

- *Gap1 (%):* is the difference between the value of the heuristic solution $T_{c\text{-}heur}$ and the optimal solution BI.

$$Gap1(\%) = \frac{T_{c\_heur} - BI}{BI} \times 100 \quad (6)$$

- *Gap2 (%):* is the deviation of the improved solution $T_{c\text{-}imp}$ from the value of the optimal solution BI.

$$Gap2(\%) = \frac{T_{c\_impr} - BI}{BI} \times 100 \quad (7)$$

B. Performance of improvement procedures

For the 160 instances, we report the experimental results with a cloud of points in Figure 5 showing the improvement percentage. Note that for almost all instances, the improvement varies between 0.72% and 10.67% and rises to 22.95% for N = 10. We can say that the improvements are significant.

In fact, for lines of mass production, the line performance is measured by its cycle time, then, even a small reduction of T_c, implies a considerable productivity gains and benefit from economies of scale. However, this reduction of T_{c_heur} is not guaranteed for any instance. We note that for some instances, the application of improvement procedures has no effect. It would be interesting to know if the heuristic solution we have is optimal.

ISSN: 2961-662X

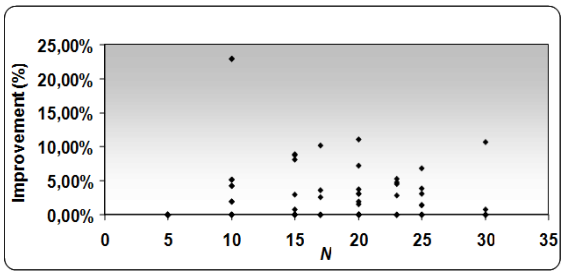

Fig. 4. Improvement percentage for different *N*

9) Deviation of the obtained solutions relative to BI:

The graphs in Fig. 5 and 6 visualize the percentage of solutions whose gap is equal to 0% (i.e the solution is optimal) and that the value of the gap is in the following intervals: from] 0%, 10%] to] 90%, 96%] with a pitch of 10%.

Fig. 5 shows the difference between the heuristic solution and the lower bound for the set of generated tests. It was found that only 2.01% (respectively 4.03% and 4.70%) cases have a *Gap1* which belongs to the interval] 90%, 96%] (] respectively 80%, 90%] and 70] %, 80%]), so for a small proportion, the heuristic solution is far from the lower bound.

In the other side, for 21.48% of the heuristic algorithm tests provide an optimal solution. It is obvious that for these 21.48% of instances, which we have given a proof of optimality, *Improvement (%)* will be zero; since T_{c_heur} reached the lower bound so the heuristic solution can't be improved furthermore. Looking at Fig. 6, we notice that 23.49% of the improved solutions are optimal. It is noted that for 8.57% of the optimal case, optimality is reached after applying improvement procedures, while the rest has already been obtained directly by the proposed heuristic.

Comparing the two Fig. 5 and 6, it is clear that the value of *Gap2* is lower than that of *Gap1*. This leads us to conclude that the application of the improvement procedures fulfill the function for which they were proposed i.e. to reduce the gap between the obtained solution and the lower bound.

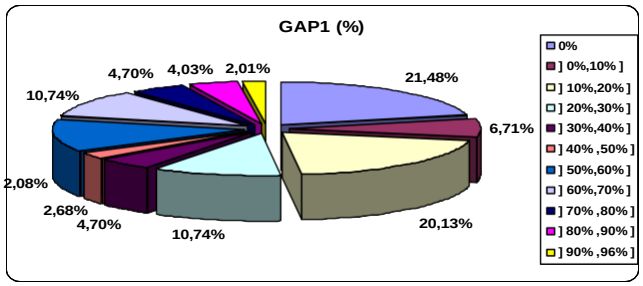

Fig. 5. Gap between S_{heur} and *BI*

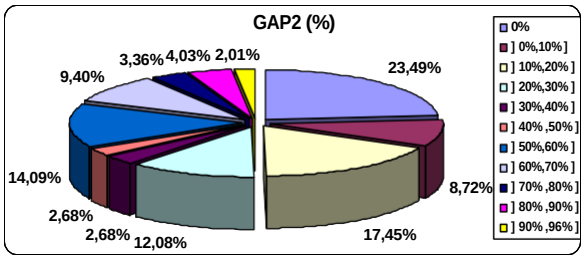

Fig. 6. Gap between *Simpr* and *BI*

C. Variation of results according to N

The performance of the proposed resolution approach is measured by the improvement percentage of the heuristic solution *Improvement (%)* and the total execution time. To evaluate the behavior of these criteria for different number of operations *N* and for the same density $D(G^r)$, we summarize the results of the tests in Tables 1, 2 and 3.

TABLE I. RESULTS ACCORDING TO *N* (LOW DENSITIES)

N	Improvement (%)		Time_execution (s)		
	Min	*Max*	*Min*	*Max*	*Moy*
5	0,00%	0,00%	13	25	21,2
10	—	22,95%	37	55	45,1
15	—	8,83%	23	77	46,9
17	—	10,14%	49	67	58,9
20	—	11,11%	46	72	60,5
23	—	5,29%	44	109	67,7
25	—	6,82%	59	112	76,9
30	—	10,67%	65	283	123,7

TABLE II. RESULTS ACCORDING TO *N* (AVERAGE DENSITY)

N	Improvement (%)		Time_execution (s)		
	Min	*Max*	*Min*	*Max*	*Moy*
5	0,00%	0,00%	12	26	17,4
10	—	22,95%	23	49	33,1
15	—	0,00%	25	54	38,4
20	—	3,08%	31	74	46,9

TABLE III. RESULTS ACCORDING TO N (HIGH DENSITY)

N	Improvement (%)		Time_execution (s)		
	Min	*Max*	*Min*	*Max*	*Moy*
5	0,00%	0,00%	12	26	17,4
10	—	22,95%	23	49	33,1
15	—	0,00%	25	54	38,4
20	—	3,08%	31	74	46,9

In Table 1, the problem size *N* varies (*N* = 5, 10, 17, 20, 23, 25, 30) for a family of low densities ($0.11 \leq D(G^r) \leq 0.2$). Tables 2 and 3 show respectively the variation of *Improvement* (%) and *Time_execution* (s) according to *N*, for an average density ($D(G^r) = 0.5$) and high density ($D(G^r) = 0.7$).

10) Evolution of Improvement (%) according to N

Note that the improvement percentage of S_{heur} doesn't depend on the number of operations to affect but rather depends on various problem constraints (precedence, exclusion for stations and blocks, inclusion for stations, m_0, n_0 and i_0),

ISSN: 2961-662X

since operations exchange between the blocks of the bottleneck station or operation exchange between the station and its adjacent stations, is possible only when all constraints are satisfied.

- ***Evolution of Time_Execution(s) according to N*:** from the three tables, we can study the *Time_execution* evolution shown in Fig. 8. It is obvious that whenever the operation number increases, the average computation time increases also, because we spend more time to allocate these operations on different stations and blocks. But it still remains low for all the 160 instances, it ranges from 15s (for N = 5 and $D\ (G^r)$ = 0.7) to 123s (for N = 30 and $D\ (G^r)$ = 0.11).

D. Variation of results according to D(Gr)

- ***Improvement evolution according to D(G^r):*** we report the evolution of the average improvement in Fig. 7. These results show that whenever $D(G^r)$ increases the improvement decreases. For high densities, the average improvement of the heuristic solution by managing the bottleneck station becomes low or zero. This behavior can be explained by the fact that a high density reflects that precedence relations aren't flexible to perform exchange of operations between the blocks of the bottleneck station or the exchange with the adjacent station.

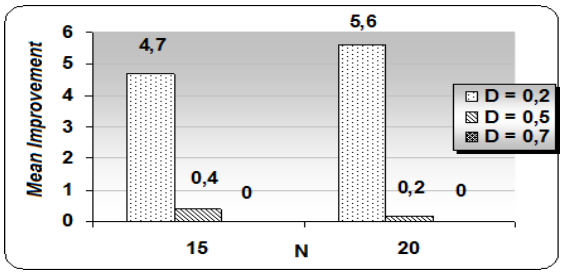

Fig. 7. Evolution of the Mean Improvement according to N and $D(G^r)$

- ***Time_execution evolution according to D(G^r):*** returning to Fig. 8, we can see the details of the time resolution behavior for series of tests with N = 5, 10, 15, 20. For the same N, we must observe the three abscissas labeled ($D(G^r)$ = 0.2, $D(G^r)$ = 0.5 and $D(G^r)$ = 0.7), which allows us to study the influence of the $D(G^r)$. We see a direct link between $D(G^r)$ and the computation time. Indeed, observing each series separately, we find a significant reduction of the execution time for a high density ($D(G^r)$ = 0.7). Thus, it can be concluded for each series, the family of tests at low density ($D(G^r)$ = 0.2) is always more difficult to solve than tests with an average density ($D(G^r)$ = 0.5). Similarly, the series with an average density are more expensive in computation time than series with the highest density. The difficulty of low-density instances can be explained by the large number of possible operations assignments to stations and blocks. So, when the density is higher, the precedence constraints become numerous and the problem becomes less flexible and the solution is obtained rapidly. However, the two parameters N and D are not the only factors that influence the computation time, especially as the difficulty in some instances, may be due to the numerical values of the input data which are generated randomly.

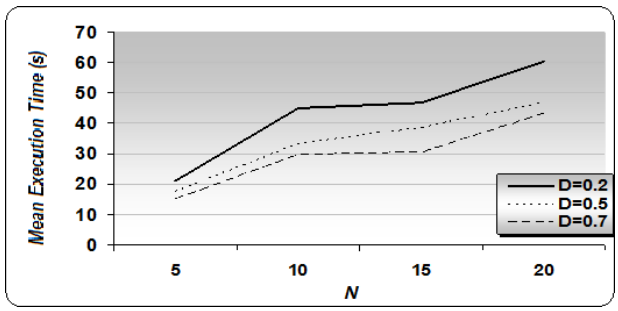

Fig. 8. Evolution of the Mean Execution Time according to N and $D(G^r)$

VII. CONCLUSION

According to the experimental results, we obtain in 23.49% of cases, an optimal solution. In addition, in 32.21% of cases, the deviation from the lower bound does not exceed 10%. Moreover, the proposed approach is able to produce a solution in a very low computation time. By managing the bottlenecks stations, reducing cycle time of the machining line is up to 22.95%. In general, the rate of good solutions obtained is interesting then we can say that our resolution approach is quite effective. The use of metaheuristics can improve more the quality of the obtained solution, and is in our opinion a logical sequence to this work.

REFERENCES

[1] O. Guschinskaya, and A. Dolgui, "A Transfer Line Balancing Problem by Heuristic Methods: Industrial Case Studies*," Decision Making In Manufacturing and services*, vol. 2. 2008. NO.1-2. pp. 33-46.

[2] H. Chehade, A.Dolgui, F. Dugardin, L. Makdessian, and F.Yalaoui, "Multi-objective Approach For Production Line Equipment Selection", *Management and Production Engineering Review*, vol. 3. Number 1. pp. 4–17, March 2012..

[3] C. Boutevin, M. Gourgand et S. Norre, "Méthodes d'optimisation pour le problème de l'équilibrage de lignes d'assemblage," *MOSIM'03*, 2003.

[4] I. Baybars, "A survey of exact algorithms for the Simple Assembly Line Balancing," *Management Science*, 32, pp. 909-932, 1986.

[5] O. Guschinskaya, et A. Dolgui, "Equilibrage des lignes d'usinage : la résolution du problème de grande taille," École Nationale Supérieure des Mines de Saint-Étienne, Rapport de recherche 2006-500-011, Oct. 2006.

[6] S. Belmokhtar, "Lignes d'usinage avec équipements standard : modélisation, configuration et optimisation," Thèse en Génie Industriel, École Nationale Supérieure des Mines de Saint-Étienne, Déc. 2006.

[7] A. Dolgui, N. Guschinsky, G. Levin and J. Proth, "Optimisation of multi-position machines and transfer lines," *European Journal of Operational Research*, 185, pp. 1375-1389, 2008.

[8] A. Dolgui, N. Guschinsky, Y. Harrath and G. Levin, "Une approche de programmation linéaire pour la conception des lignes de transfert," in *Proc. MOSIM'01*, 2001, pp.353-361.

[9] A. Dolgui, A et I. Ihnatsenka, " Lignes d'usinages avec têtes multibroches : un nouveau problème d'optimisation," *MOSIM'06*, 2006.

[10] A. Dolgui, , N. Guschinsky and G. Levin, "Enhanced mixed integer programming model for a transfer line design problem*," Computers & Industrial Engineering* 62, pp. 570–578, 2012.

ISSN: 2961-662X