Comparison of Metaheuristics and Exact Method for the Dynamic Line Rebalancing Problem

Mohand Lounes Bentaha, Salma El Abdellaoui

Comparison of Metaheuristics and Exact Method for the Dynamic Line Rebalancing Problem

M.-Lounes Bentaha[1][0000-0001-6564-3435] and Salma El Abdellaoui[2]

[1] Université de Lyon, Université Lumière Lyon 2, INSA Lyon, Université Claude Bernard Lyon 1, DISP, EA4570, 69676 Bron, France
[2] OMP France, 20 Boulevard Montmartre, 75009 Paris, France
mohand.bentaha@univ-lyon2.fr
salma.el.abdellaoui.8@gmail.com

Abstract. Assembly lines are the most widely used systems for industrial mass production. A main objective in such a system is to ensure a workload balancing among its workstations and optimize it at the operational level. However, this balancing is affected by various disturbances which induce delays and then generate additional costs and deteriorate the performance of the assembly line. To remedy the negative effects of such disturbances, methods allowing real-time rebalancing are needed. The problem is known as the dynamic rebalancing of assembly lines. This work proposes a comparative study of three metaheuristics performances in solving this problem, namely: Iterated Local Search, Genetic Algorithm and Filters Beam Search-Ant Colony Optimization. The choice of these metaheuristics is motivated by their reputation for quickly and efficiently solving assembly line balancing problems. An exact method, whose performance is compared to the three selected metaheuristics, is also considered. The four approaches are applied to instances of industrial size and complexity known in the assembly line balancing literature. This benchmark data set guarantees coverage of almost all cases that an industrial could encounter. Obtained results showed the metaheuristics efficiency in solving large instances and that the exact method is recommended for small ones. Efficiency is measured here in terms of resolution speed (few seconds are required) and the quality of returned rebalancing solution. A rebalancing solution is of good quality if its cycle time is less than or equal to the initial line takt time.

Keywords: Assembly, Assembly line, Line balancing, Dynamic rebalancing, Reconfiguration, Metaheuristics, Exact method, Disturbance, Uncertainty.

1 Introduction

At the operational level, the assembly process can be affected by various disruptions such as unscheduled shutdowns, breakdowns, repairs, stockouts, etc. These disturbances cause delays which affect the initial balancing of the line by generating additional costs and deteriorating its performance. The objective is therefore to remedy the negative effects of these disturbances by techniques that allow real-time workload

rebalancing of all workstations. This problem is known as the dynamic rebalancing of assembly lines [1]. In this article, the seek for a line rebalancing is based on a reassignment of tasks to workstations to absorb, as possible, the delay induced at a given time. Operators are therefore assumed to be versatile and can easily adapt to changes in the assembly line. An industrial case, in the Trane company, which includes such a possibility is described in [2]. This study finds its application particularly at assembly lines level managed by a pull system. To synchronize the flows of these lines, it is important that all operate at a same rate defined by the Takt time (T_t), see Fig. 1 for an illustration. This will ensure balanced lines that meet customer demand. The takt time is the rate at which it is needed to complete the production process to meet customer demand, i.e. T_t = net available work time / customer demand.

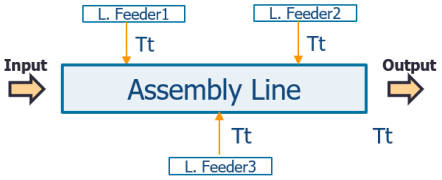

Fig. 1. Example of assembly lines dedicated to a family of products.

Real-time rebalancing can only be possible with fast algorithms (a response time of few seconds). In this article, a rebalancing solution is said to be efficient if it is obtained quickly (in a few seconds) and that the returned Cycle time (C_t) is less than or equal to the takt time. Cycle time is the time it takes to complete the production of one unit from start to finish, i.e. C_t = net available work time / number of units produced. Takt time is based on customer demand whereas cycle time is work process based [3].

To identify methods that can efficiently solve the problem defined above, a literature review is carried out. Considered articles are only those dealing with balancing, rebalancing, and dynamic rebalancing of assembly lines. Most studies address the problem of initial balancing of assembly lines (long-term decision) [4, 5]. A decision that takes place at the design and implementation phase of the assembly line. A recent trend is to deal with the rebalancing problem [3, 6–9] to cope with the dimensional changes of the market due to seasonality and the life cycle of products for example and structural changes linked to line reconfiguration and layout (add or remove workstations, etc.). However, the problem of dynamic rebalancing of assembly lines is rarely studied as such [1, 10]. It is a real-time decision aid whose objective is to remedy the various disturbances that affect the line. For these three problems, different formalizations based on mathematical programming are proposed to model them. To solve balancing and rebalancing problems, a wide variety of exact, heuristic and metaheuristic methods are proposed. But, for dynamic rebalancing we found mainly two approaches: ILS (Iterated Local Search) [1] and Communicating Automata approaches that are applied to small instances [11]. To identify the potential metaheuristics to be selected for our study, an additional work on the classification of the latter is carried out by analysing the studies in [12–15]. Two main classifications exist : the one

based on [15]: Local Search Metaheuristics, Constructive Metaheuristics, Population-based Metaheuristics and Hybrid Metaheuristics; the one based on [13]: Single-Solution Metaheuristics (SSM) and Population Metaheuristics (PM). In addition, in this latter classification, in each category we distinguish between essentially constructive metaheuristics (Primarily Constructive, PC) and evolutionary metaheuristics (Improvement Metaheuristics, IM). The combination of these two classifications made it possible to establish the classification of metaheuristics represented in Fig. 3.

2 Problem Modeling and Solution Approaches

The studied system is assembly lines. An assembly line is made up of workstations arranged sequentially. The workload of each workstation is defined as the sum of the operating times of the tasks assigned to it. An assembly line is perfectly balanced if the workloads of all stations are the same and equal to cycle time. It is said to be balanced if the differences in workloads between the stations are as close as possible. If moreover the cycle time is less than or equal to the takt time, then such a line will satisfy the customer demand (see Fig. 2, top). A line is affected when disturbing elements occur at a time T_0. In this case, the assembly line is partitioned into two parts: a fixed one containing completed tasks and a dynamic one containing the unrealized tasks that require dynamic balancing (Fig. 2, bottom). The objective is to find a dynamic reassignment of non-performed tasks to the corresponding workstations that allows balancing of the remaining work, respecting the precedence constraints among tasks and, as possible, absorbs the induced delay (i.e. ensure as possible the cycle time value to be less than or equal to the takt time). The mathematical program **LPDR** models the problem described above [1].

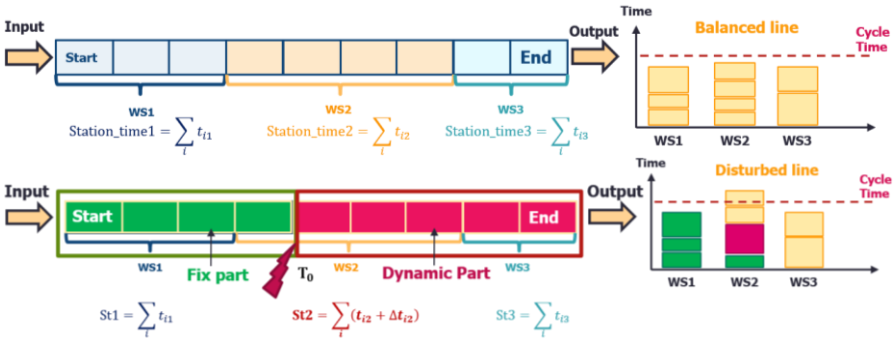

Fig. 2. Illustration of balanced and disturbed line concepts.

$$\min \max_{j \in J^*} \left\{ \mathrm{St_j} = \sum_{i \in I^* \cup A_{j_0}} (t_i + \Delta t_i)\, x_{ij} \right\} \quad (\mathbf{LPDR})$$

$$\mathbf{Subject\ to}: x_{ij_0} = 1, \forall\, i \in A_{j_0} \setminus I^* \ (1)$$

$$x_{ij} = 0, \forall\, i \in A_{j_0} \setminus I^*, \forall\, j \in J^* \setminus \{j_0\} \ (2)$$

$$\sum_{j \in J^*} x_{ij} = 1, \forall\, i \in I^* \quad (3)$$

$$\sum_{j \in J^*} j\, x_{i'j} \leq \sum_{j \in J^*} j\, x_{ij}, \forall\, i \in I^*, \forall\, i' \in P_i \quad (4)$$

$$x_{ij} \in \{0,1\}, \forall\, i \in I^* \cup A_{j_0}, \forall\, j \in J^* \quad (5)$$

where, n: number of tasks, $n \in \mathbb{N}^*$; m: number of workstations, $m \in \mathbb{N}^*$; I: set of all tasks, $I = \{1,2,\dots,n\}$; J: set of all workstations, $J = \{1,2,\dots,m\}$; i: a single task, $i \in I$; j: a single workstation, $j \in J$; I^*: set of tasks which need to be re-assigned for re-balancing, $I^* \subseteq I$. Note that task impacted by the disturbance at time T_0 (denoted i_0) is an element of I^*; J^*: set of workstations where tasks in I^* can be reassigned, $J^* \subseteq J$; j_0 denotes the workstation which is impacted by the disturbance at time T_0 ($j_0 \in J^*$); $(A_j)_{j \in J}$: initial assignment of tasks to workstations J (before the disturbance at time T_0 occurs); $(A_j)_{j \in J}$ forms a partition of I; $(P_i)_{i \in I}$: sets of all predecessors of tasks I; T_t: takt time, $T_t > 0$; C_t: cycle time, $C_t > 0$; t_i: processing time of a task $i \in I$, $t_i > 0$; Δt_i: delay of a task $i \in I$, $\Delta t_i \geq 0$; x_{ij}: decision variable, $x_{ij} = 1$ if task $i \in I$ is assigned to workstation $j \in J$, $x_{ij} = 0$ otherwise.

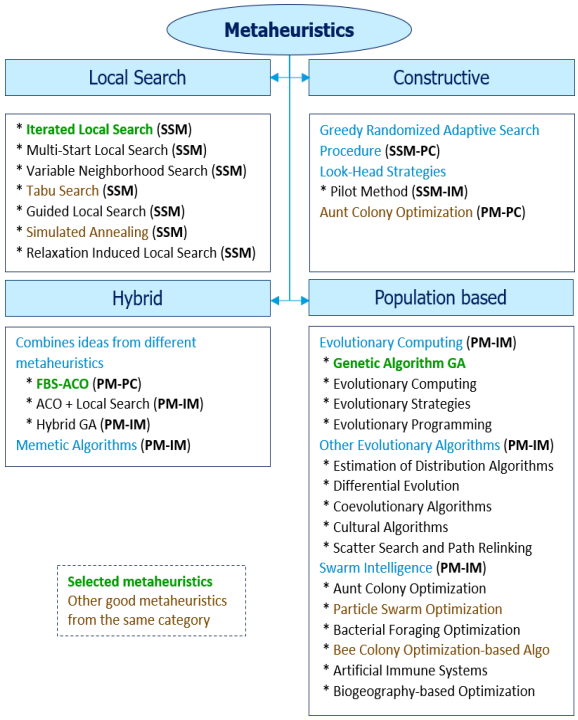

Fig. 3. Classification of metaheuristics and selection of used ones.

To solve the **LPDR** problem, our choice is based on three metaheuristics, namely: Iterated Local Search (ILS), Genetic Algorithm (GA) and Filters Beam Search-Ant Colony Optimization (FBS-ACO), see Fig. 3. The choice of these metaheuristics is based on an analysis of the literature on balancing and rebalancing of assembly lines.

This analysis focused on metaheuristics whose effectiveness meets the criteria of speed and quality of cycle time minimization. The exact algorithm selected here is the default CPLEX Mixed Integer Programming (MIP) optimizer (based on branch & cut) [16]. No constructive metaheuristics are selected because of their penalizing computational time which increases significantly by increasing the number of tasks [17]. ILS is chosen based on its performance reported in [1]. GA is chosen among the population algorithms because it is the most widely used and has proven its efficiency and speed in solving problems of balancing and rebalancing [6, 17]. Finally, for hybrid metaheuristics, FBS-ACO is chosen thanks to its verification of the two previous criteria. Indeed, FBS-ACO is an improved version of the ACO algorithm from which it inherited the efficiency, and the FBS filters have allowed a significant improvement in computation time [8]. Brief pseudocodes of the three metaheuristics are given hereafter. ILS consists in applying a local search to a unique solution and a disturbance mechanism in several iterations, see [1]; in GA, an individual in the population represents a solution. This algorithm consists of building populations by improving their individuals from one generation to another using mutation, crossover and selection, see [18, 19]; FBS-ACO consists of using the characteristics of ant colony algorithm to build a solution respecting the defined number of workstations. In addition, the application of local and global evaluations of FBS makes it possible to select at each stage the next destination of an ant to build a feasible solution [18].

Algorithm 1: ILS

Generate initial line balancing S_0
$S \leftarrow S_0$
$S'' \leftarrow S_0$
do
  $S' \leftarrow$ LocalSearch(S'')
  if ($C_t(S') < C_t(S)$) **then**
    $S \leftarrow S'$
  end if
  $S'' \leftarrow$ Disturb(S');
while (IterCount<maxIter && $C_t(S) > T_t$)

Algorithm 2: GA

Generate initial population considering the initial line balancing
Compute fitness of initial population individuals
Select the best individual
while (IterCount<maxIter && $C_{t_{population}}(S) > T_t$) **do**
  Apply elitism
  Apply crossover
  Apply mutation
end
Evaluate the best fitness of all generations
Select the best among the best individuals

Algorithm 3: FBS-ACO

```
Save the number of workstations of the initial line balancing
Initialise pheromone quantity
Generate initial Beam nodes
for (k from 0 to Ants number) do
    Assign fixed tasks of impacted workstation j_0
    Assign Beam node k
    while (some tasks are not assigned || workstations number is reached) do
        Identify candidates to be assigned using global evaluation
        Select best task candidate using local evaluation
        Update locally the pheromone
    end
    Update globally the pheromone
end
```

3 Numerical Experiments and Performance Comparison

The exact method (default CPLEX MIP optimizer), ILS, GA and FBS-ACO metaheuristics are implemented in Linux using C++ on a PC of 8Go RAM and 2.30 GHz CPU. Data set of 28 benchmark instances known in the assembly line balancing literature are used, see https://assembly-line-balancing.de/. Size and complexity of such instances guarantees coverage of almost all situations that an industrial could encounter. In addition, to cover most situations of line disturbances, we defined a design of experiments based on three main criteria, as follows: a disturbance can occur at the beginning or middle of the line; a disturbance can be low, average or high; task processing times can be high, average (original values) or low. Values of delays and task processing times are generated after analysis of total initial balancing idle time, number of tasks, number of workstations and initial cycle time. To generate an initial balancing for each instance, we have used SALOME [20]. So, each approach has to solve $28 \times 2 \times 3 \times 3 = 504$ different instances. To simplify the understanding and analysis of the obtained results, we defined three categories of the 504 instances: category 1 composed of instances whose tasks number doesn't exceed 50 (252 instances); in category 2 tasks number is greater than 50 but doesn't exceed 100 (198 instances); in category 3 tasks number exceeds 100 (54 instances). Resolution time of the exact method is limited to one hour.

The performance of each method is measured using the number of returned efficient solutions (takt-time respected and speed of resolution). Note that a solution is inefficient if the returned cycle time is not less than or equal to the takt time or if the resolution time is large (case of the exact method in particular). Analysis of the results consists in finding the number of solved instances (efficient solutions) by each metaheuristic and identifying their characteristics. Instances for which a metaheuristic has not found an efficient solution doesn't mean that really an efficient solution doesn't exist unless, the exact method has failed to solve it efficiently. The results have shown that for some instances the delay can be absorbed immediately, without doing any effort. This is the case when the cycle time of the line considering the delay

is less than or equal to the takt time. It was the case for 73 instances (47 of categoty 1 and 26 of category 2). For the remaining 431 instances, the delay can't be absorbed initially, hence an efficient solution could be found using one of the four solution approaches.

For exact method, analysis is based on instances of category 1 since it is the only one where all instances (205) are solved efficiently. Delay was not absorbed for 79 instances and was absorbed for the remaining 126 instances. Most of the cases where delay is absorbed are characterised with low or average task times, see Table 1- exact method; 123 instances are solved in an average time of 0.56 seconds each.

Table 1. Characteristics of solved instances and solution methods performances.

		Solved	Tt respected		Dyn tasks nbr			Dyn workstations nbr			Tasks/workstations			Excecution time (s)		
					median		mean	median		mean	median		mean	median		mean
Exact M. Category 1	Beginning	205	69	126	29	28	24	6	6	5	4,28	4,14	4,13	0,34	0,039	35,33
	Middle		57		14			3			4			0,025		
ILS Category 1	Beginning	205	22	63	31	16	19	6	4	4	5,33	4,33	4,66	0,0006	0,0003	0,0004
	Middle		41		13			3			4			0,0002		
ILS Category 2	Beginning	172	16	27	70	58	61	15	12	15	4,75	5,38	5,76	0,0041	0,0022	0,0033
	Middle		11		47			7			7,42			0,0007		
ILS Category 3	Beginning	54	1	3	111	59	76	15	8	10	7,37	7,37	7,38	0,0114	0,0024	0,0049
	Middle		2		59			8			7,37			0,0017		
					mean		32	mean		7	mean		4,66	mean		0,0012
		Solved	Tt respected		Dyn tasks nbr			Dyn workstations nbr			Tasks/workstations			Excecution time (s)		
					median		mean	median		mean	median		mean	median		mean
GA Categoty 1	Beginning	205	25	73	28	15	17	4	3	3	5,33	4,33	4,7	0,483	4,33	0,429
	Middle		48		14			3			4,33			0,109		
GA Categoty 2	Beginning	172	2	11	53	47	46	4	4	4	13,25	11,75	11,93	0,0041	11,75	0,869
	Middle		9		47			4			11,75			0,0007		
GA Categoty 3	Beginning	54	0	0	0	0	0	0	0	0	-	-	-	-	-	-
	Middle		0		0			0			-			-		
					mean		26	mean		4	mean		5,66	mean		0,486
		Solved	Tt respected		Dyn tasks nbr			Dyn workstations nbr			Tasks/workstations			Excecution time (s)		
					median		mean	median		mean	median		mean	median		mean
FBS-ACO Categoty 1	Beginning	205	67	87	29	28	24	6	6	5	4,28	4,16	4,73	0	0	0,0016
	Middle		20		14			3			3,62			0		
FBS-ACO Categoty 2	Beginning	172	55	68	83	75	71	13	13	14	7,23	7,23	6,53	0,021	0,021	0,0224
	Middle		13		41			6			8			0,011		
FBS-ACO Categoty 3	Beginning	54	18	21	148	148	163	15	15	19	7,4	7,4	9,68	0,254	0,2545	0,3805
	Middle		3		144			21			6,85			0,4		
					mean		65	mean		11	mean		6,07	mean		0,058

ILS solved 93 instances out of 431, see characteristics in Table 1- ILS. GA solved 84, Table 1- GA and FBS-ACO solved 176 instances, Table 1- FBS-ACO. To summarize, GA is efficient when the number of tasks and workstations are low. ILS is efficient for small to medium sized instances and FBS-ACO is efficient when the delay occurs at the beginning of the line. Exact method is efficient for small sized instances (no more than 50 tasks). Among all efficiently solved instances only 17 are common to the four solution approaches. These 17 instances are used to compare performance of ILS, GA and FBS-ACO to the exact method. The ratio C_t/T_t is used as comparison criteria, see Fig. 4. It is seen that the three metaheuristics gave solutions very close to the optimal ones. FBS-ACO and GA found the optimal solution for 6 of the 17 instances and ILS found only 2. It is also shown that when the dimension of the assembly line increases, GA finds solution with good quality. However, for most instances, ILS gives an acceptable solution but not close to the optimum compared to the other metaheuristics.

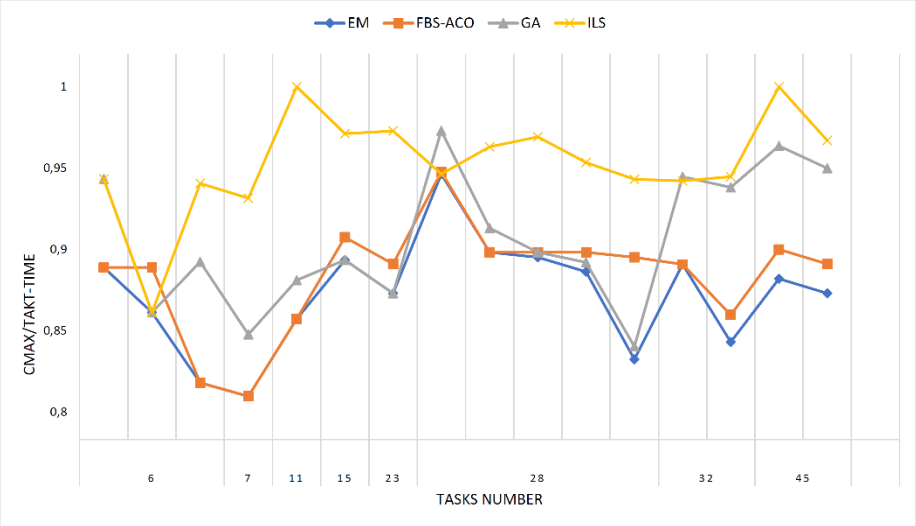

Fig. 4. Performance comparison of solution approaches based on C_t/T_t values.

4 Conclusion and Recommendation

In this study, a performance comparison of three metaheuristics (ILS, GA, FBS-ACO) and exact method for the dynamic line rebalancing problem is conducted. The metaheuristics generated satisfactory results compared to the exact method. Obtained results analysis allowed to identify the most suitable method to apply for each line disturbance situation. GA are efficient on small instances with an average number of workstations equal to four. ILS can be used for small to medium size instances. FBS-ACO is more efficient for assembly lines that have many workstations. Finally, the exact method is very efficient for small instances (no more than fifty tasks).

As a future research development, we will investigate a fifth solution approach based on Artificial Intelligence techniques namely Machine Learning although FBS-ACO already can be seen as an AI approach since it is multi-agent based inspired by the behavior of real ants.

References

1. Antoine, M., El-Haouzi Hind, B., Cherif-Khettaf Wahiba, R., Mohand Lounes, B.: Iterated Local Search for dynamic assembly line rebalancing problem. IFAC-PapersOnLine. 49, 515–519 (2016). https://doi.org/https://doi.org/10.1016/j.ifacol.2016.07.679
2. El Haouzi, H.: Approche méthodologique pour l'intégration des systèmes contrôlés par le produit dans un environnement de juste-à-temps: Application à l'entreprise TRANE, https://tel.archives-ouvertes.fr/tel-00362316, (2008)

3. Oliveira, F.S., Vittori, K., Russel, R.M.O., Travassos, X.L.: Mixed assembly line rebalancing: A binary integer approach applied to real world problems in the automotive industry. Int. J. Automot. Technol. 13, 933–940 (2012)
4. Micieta, B., Stollmann, V.: Assembly line balancing. DAAAM Int. p. 257–264 (2011)
5. Sivasankaran, P., Shahabudeen, P.: Literature review of assembly line balancing problems. Int. J. Adv. Manuf. Technol. 73, 1665–1694 (2014)
6. Yang, C., Gao, J., Sun, L.: A multi-objective genetic algorithm for mixed-model assembly line rebalancing. Comput. Ind. Eng. 65, 109–116 (2013). https://doi.org/https://doi.org/10.1016/j.cie.2011.11.033
7. Celik, E., Kara, Y., Atasagun, Y.: A new approach for rebalancing of U-lines with stochastic task times using ant colony optimisation algorithm. Int. J. Prod. Res. 52, 7262–7275 (2014). https://doi.org/10.1080/00207543.2014.917768
8. Zha, J., Yu, J.: A hybrid ant colony algorithm for U-line balancing and rebalancing in just-in-time production environment. J. Manuf. Syst. 33, 93–102 (2014). https://doi.org/https://doi.org/10.1016/j.jmsy.2013.08.002
9. Gamberini, R., Gebennini, E., Grassi, A., Regattieri, A.: A multiple single-pass heuristic algorithm solving the stochastic assembly line rebalancing problem. Int. J. Prod. Res. 47, 2141–2164 (2009). https://doi.org/10.1080/00207540802176046
10. Zhiyuan, Z., Jie, T., Wancheng, N., Yiping, Y.: A multi-objective dynamic rebalancing scheduling algorithm for mixed-model assembly line. In: 2012 6th International Conference on New Trends in Information Science, Service Science and Data Mining (ISSDM2012). pp. 586–591 (2012)
11. Antoine, M., Hind, B.E.-H., André, T., Jean-François, P.: Dynamic Rebalancing of an Assembly Line with a Reachability Analysis of Communicating Automata. In: Grabot, B., Vallespir, B., Gomes, S., Bouras, A., and Kiritsis, D. (eds.) Advances in Production Management Systems. Innovative and Knowledge-Based Production Management in a Global-Local World. pp. 597–604. Springer Berlin Heidelberg (2014)
12. Boussaïd, I., Lepagnot, J., Siarry, P.: A survey on optimization metaheuristics. Inf. Sci. (Ny). 237, 82–117 (2013). https://doi.org/https://doi.org/10.1016/j.ins.2013.02.041
13. Gendreau, M., Potvin, J.-Y.: Metaheuristics in combinatorial optimization. Ann. Oper. Res. 140, 189–213 (2005). https://doi.org/10.1007/s10479-005-3971-7
14. Osman, I.H., Laporte, G.: Metaheuristics: A bibliography. Ann. Oper. Res. 63, 511–623 (1996). https://doi.org/10.1007/BF02125421
15. Sörensen, K., Glover, F.W.: Metaheuristics. In: Gass, S.I. and Fu, M.C. (eds.) Encyclopedia of Operations Research and Management Science. pp. 960–970. Springer US (2013)
16. IBM ILOG CPLEX: User's Manual for CPLEX. (2021) URL: https://www.ibm.com/docs/en/icos/12.7.1.0?topic=cplex-users-manual
17. Mcgovern, S.M., Gupta, S.M.: Combinatorial optimization analysis of the unary NP-complete disassembly line balancing problem. Int. J. Prod. Res. 45, 4485–4511 (2007). https://doi.org/10.1080/00207540701476281
18. Gendreau, M., Potvin, J.-Y., others: Handbook of metaheuristics. Springer (2010) https://doi.org/10.1007/978-1-4419-1665-5
19. REEVES, C. ed.: Modern Heuristics Techniques for Combinatorial Problems. Nikkan Kogyo Shimbun. 1–320 (1997) https://ci.nii.ac.jp/naid/10010555508/en/
20. Scholl, A., Klein, R.: SALOME: A Bidirectional Branch-and-Bound Procedure for Assembly Line Balancing. INFORMS J. Comput. 9, 319–334 (1997)