Mail sorting optimization: a case study of the french postal company La Poste

Emmanuelle Amann * **Evgeny Gurevsky** ** **Nasser Mebarki** **

* *La Poste, Nantes, France*
** *LS2N, Université de Nantes, France*

Abstract: This paper addresses a mail sorting optimization problem encountered by a major French mail delivery provider La Poste. To simplify the operators' handling during the sorting process, it is necessary to balance the mail flow between the outputs of each sorting machine. This problem can be considered as a variant of the well-known assembly line balancing problem (ALBP) with a particular structure of precedence constraints. At first, to handle efficiently small- and medium-size academic instances of this problem, we propose a mixed-integer linear programming (MILP) model aiming to minimize the difference between the most and least loaded outputs. Finally, to deal successfully with real industrial large-size instances, we develop an efficient heuristic, inspired by simulated annealing, which focuses on minimizing the non-linear standard deviation between the output loads. Both approaches have demonstrated a very good overall performance for the respective instance categories. Computational results are reported as well.

Keywords: Mail sorting, balancing, smoothness, ALBP, MILP, heuristic, simulated annealing, industrial case.

1. INTRODUCTION

All over the world, incumbent mail delivery operators are facing a drastic decrease in the volume of mail exchanged. In France, La Poste, a public mail distribution operator, has seen the volume of mail fall from 18 billion in 2008 to 9 billion in 2018, with forecasts of 5 billion in 2025. To cope with this problem, La Poste intends to build a dynamic organization of mail sorting to optimize the use of sorting machines.

Currently, to facilitate postmen's rounds, the batch of mail assigned to them is sorted in the order of geographical location of mailboxes and stored in containers. Due to the limited number of outputs of the sorting machines, sorting is carried out in two steps. During the first sorting step, the containers can be filled with mail from different rounds, while at the end of the second step, each container corresponds to a single round. At present, this sorting process is done with a static view based on the day of the week and not on the actual mail flow. This leads to unbalanced containers on the first step, with some full of mail and others almost empty. To simplify the operators' handling, it would be of great interest to obtain the load between containers as smoothly as possible at the end of the first step while satisfying the mail precedence constraints, linked to the geographical location of the mailboxes. This problem can be considered as a variant of the well-known assembly line balancing problem (ALBP) with a fixed number of workstations. This latter generally consists of distributing a given set of tasks (resp. mail batches in our case) to a given set of workstations (resp. containers) while satisfying some technological restrictions and optimizing certain production objectives. Among the restrictions used, one can notice task precedence (resp. mail precedence) and cycle time constraints. The first constraints are usually represented as a directed acyclic weakly connected graph, in which the nodes correspond to the tasks and the arcs express their partial execution order. The cycle time constraints are not mandatory. They determine a certain limit on the workload of each workstation that can not be exceeded. Concerning the production objectives, we can mention the following ones: the minimization of the workload of the busiest workstation, the maximization of the workload of the least busy workstation, or the optimization of certain smoothness indexes (SIs) between the workstation workloads. For more details, we refer the reader to the three well-known ALBP reviews as Boysen et al. (2007), Battaïa and Dolgui (2013), and Boysen et al. (2022).

There are two main differences between the ALBP and the problem studied in this article. The first difference consists in a particular structure of the mail precedence constraints, which are represented as a set of independent chains of the strict delivery order of the mail batches. Each chain corresponds to a certain postmen's round. The second one concerns the study of two new SIs: the difference between the most and least charged container loads, denoted f_1, and the standard deviation between the container loads, denoted f_2. In addition, to the best of our knowledge, ALBP variants have never been studied in the context of mail sorting.

The remainder of this paper is organized as follows. Section 2 presents a short literature review of ALBPs with a fixed number of workstations and related to the study of SI. Section 3 describes some details of La Poste's mail sorting. Section 4 provides a mixed-integer linear programming (MILP) model for the problem addressed

at La Poste that aims to minimize the linear objective f_1. Section 5 outlines an efficient heuristic, inspired by simulated annealing, which focuses on minimizing the non-linear objective f_2 for the same problem. Finally, Section 6 reports and analyzes the experimental results of the MILP model and the heuristic, performed on small and large size instances of La Poste.

2. RELATED WORKS

To the best of our knowledge, Rachamadugu and Talbot (1991) were historically the first authors introducing and studying several smoothness indexes (SIs) such as workload variance (WV) and mean absolute deviation (MAD) to be minimized for ALBPs. Here, WV is expressed as the average squared deviation of the real workloads from the *ideal* one, which is computed as the sum of the processing time of all tasks divided by the number of workstations. As concerns MAD, it is similar to WV, where the squared values are replaced by the absolute ones. Despite discussing these two SIs, the authors proposed a two-stage heuristic method only for a single MAD objective. For the same objective, Kim et al. (1998) proposed an evolutionary algorithm. Kim et al. (1996) was the first paper proposing a solution procedure (a genetic algorithm) for an ALBP with the WV objective. Pinnoi and Wilhelm (1997) develop a branch-and-cut procedure for an ALBP aiming to maximize the minimal workstation workload. Azizoğlu and İmat (2018) developed the first exact solution approach (task-oriented branch-and-bound) for an ALBP with a quadratic smoothness index aiming to minimize the sum of squared workloads.

It is also important to mention that the considered SIs have also been studied in multi-objective frameworks for ALBPs. For example, Nearchou (2008) and Otto and Scholl (2011) investigate SIs in a weighed sum of several objectives. Pastor (2011) and Eswaramoorthi et al. (2012) incorporate them as secondary objectives in the lexicographic approach. Finally, Nearchou (2011) explores one of SIs as a primary objective aimed at finding a set of non-dominated solutions in the Pareto sense.

It is relevant to note that the SIs have also been addressed in the context of scheduling with parallel machines/processors (see, *e.g.*, Ho et al., 2009; Ouazene et al., 2014). Develop this part.

3. LA POSTE MAIL SORTING

As mentioned in Section 1, mail sorting at La Poste is done in two steps because of the limited number of outputs for the sorting machines. To avoid the large disparity in output loads at the end of the first step and, therefore, to simplify operators' handling, it is important to conduct a preliminary analysis of all involved postmen's rounds before starting the sorting process. This analysis can help, among others, to determine the outputs for each mail batch in advance such that the load of each output becomes as equal as possible. This latter represents the optimization problem handled in this paper.

At La Poste, each postmen round is considered as a set of so-called distribution points (DPs), which are strictly ordered between them with respect to a geographical location of mailboxes. The number of DPs per round can be different from one round to another. Each DP is strictly associated with a particular unique address. However, the number of mails affiliated with this address can be substantial, constituting what is known as a batch of mail attached to that DP. The total thickness of a mail batch is also called as the batch volume.

Continue to check.

During the sorting process, each DP is attributed to an output of the machine. For example, Table 1 shows an example of four rounds with five outputs. In this example, each DP is attributed sequentially to each output, i.e., the first DP to the first output, the second DP to the second output, etc. It leads to an unbalanced configuration where the total load of the first output is 34, whereas the total load of the final output is only 4.

Round	Out. 1	Out. 2	Out. 3	Out. 4	Out. 5
1	10	7	3	8	4
2	5	6	1	7	
3	8	9	3		
4	11	4	3		
$\sum$	34	26	10	15	4

Table 1. An example of allocation of DPs to outputs with $f_2 = 12.13$

To balance the total loads of the outputs, for each round, we need to move the load of a DP to an empty cell, i.e. an empty output in the same row. For example, for round 2, the fourth DP can be moved to output #5, the only one in the same row, which is empty. However, the precedence constraints between DPs must be respected. For round 3, a feasible configuration could be: first DP in output #1, second DP in output #3 and third DP in output #5 leading to the configuration in Table 2. For round 3, a configuration where the first DP is in output #1, the second DP in output #5 and the third DP in output #3 is not feasible because the second DP must always precede the third. Nevertheless, as the precedence constraints are circular, another feasible configuration for round 3 could be: third DP in output #1, first DP in output #2 and second DP in output #3.

Round	Out. 1	Out. 2	Out. 3	Out. 4	Out. 5
1	10	7	3	8	4
2	5	6	1		7
3	8		9		3
4	11	4	3		
$\sum$	34	26	10	15	4

Table 2. The same example as in Table 1 with another allocation of DPs to outputs: $f_2 = 9.71$

4. MILP FORMULATION

In this section, for the studied problem, we present a MILP formulation, which aims at minimizing the linear objective f_1. We first introduce some useful notations, then describe the decision variables used and finally explain the MILP model with the necessary statements for the objective function as well as for any constraint.

Notations:

- $R = \{1, \ldots, |R|)\}$ is the set of all postmen's rounds. $|R|$ is also served as the number of outputs, used in the second mail sorting step;
- $B^{(r)} = \{1, \ldots, |B^{(r)}|)\}$ is the set of mail batches for the round $r \in R$;
- $v_j^{(r)}$ is the volume of the j-th mail batch in the round $r \in R$. Here, $j \in B^{(r)}$.
- $O = \{1, 2, \ldots, |O|\}$ is the set of outputs for the first mail sorting step;
- $O_j^{(r)} = \{j, j+1, \ldots, |O| - |B^{(r)}| + j\}$ is the interval of potential outputs for the j-th mail batch in the round $r \in R$;
- $U_k^{(r)} = \{j \in B^{(r)} : k \in O_j^{(r)}\}$ is the set of mail bathes of round r, which can be potentially assigned to the output $k \in O$.

Variables:

- $x_{jk}^{(r)}$ is equal to 1 if the j-th mail batch of round r is assigned to the output $k \in O_j^{(r)}$, 0 otherwise;
- $L_{\min} \geq 0$ is the minimal load per output;
- $L_{\max} \geq 0$ is the maximal load per output.

Model:

$$\min \quad f_1 := L_{\max} - L_{\min} \tag{1}$$

subject to:

$$\sum_{k \in O_j^{(r)}} x_{jk}^{(r)} = 1, \quad \forall j \in B^{(r)}, \quad \forall r \in R \tag{2}$$

$$\sum_{j \in U_k^{(r)}} x_{jk}^{(r)} \leq 1, \quad \forall k \in O, \quad \forall r \in R \tag{3}$$

$$\sum_{k \in O_j^{(r)}} k \cdot x_{jk}^{(r)} \leq \sum_{k \in O_{j+1}^{(r)}} k \cdot x_{j+1,k}^{(r)}, \quad \forall j \in B^{(r)} \setminus |B^{(r)}|, \quad \forall r \in R \tag{4}$$

$$L_{\min} \leq \sum_{r \in R} \sum_{j \in U_k^{(r)}} v_j^{(r)} \cdot x_{jk}^{(r)} \leq L_{\max}, \quad \forall k \in O \tag{5}$$

$$x_{jk}^{(r)} \in \{0, 1\}, \quad \forall j \in B^{(r)}, \quad \forall k \in O_j^{(r)}, \quad \forall r \in R$$

$$L_{\min}, \; L_{\max} \geq 0$$

The principal goal of objective function (1) is to find a solution minimizing the difference between the most and least loaded outputs. Constraints (2) require that each mail batch j of round r be assigned to exactly one output. Inequalities (3) show that, for a given round, there can be at most one mail batch per output. The precedence mail constraints for each round r are modeled by (4). Constraints (5) express the total load for each output, which can not be less than $L_{\min}$ and greater than $L_{\max}$.

5. HEURISTIC APPROACH

To consider the actual mail flow, La Poste will do the sorting mail process several times a day. Therefore, there is a need for an effective method to balance the output loads in the first pass within a very short response time, i.e., less than a minute. We first tested two metaheuristics, a simulated annealing algorithm and a taboo search algorithm. They gave good results but in a response time of several minutes, which was not acceptable for industrial use. So, we proposed a heuristic inspired by the simulated annealing method. It gave very good results on industrial instances within an acceptable response time (see Section 6). As shown in Algorithm 2, this heuristic comprises three phases. At each step of the first phase, the heuristic tries to improve the objective function. At the end of this phase, the best solution obtained is frequently a local minimum of the objective function. To improve this solution, during the second phase, the heuristic keeps solutions that degrade the objective function with a tolerance decreasing when the objective function is not improved. Finally, the third phase, similar to the first, starts with the best solution obtained at the end of the second phase. The tolerance and the decrement step used in the second phase are parameters that can be set for each configuration. After numerous experiments on real instances, the tolerance value has been set to 5% of the objective function at the end of the first phase and the decrement step to 0.02. Subsection 5.1 explains how a new solution is computed.

5.1 The neighborhood procedure

The *Neighborhood* procedure aims to balance loads of the outputs, row by row. As shown in Algorithm 1, the idea is to move loads from the most to the least loaded admissible output, while respecting the precedence constraints. An admissible output is an output in the same row with no load. If this move gives an acceptable solution, i.e., a solution improving the objective function or degrading it, considering the tolerance, we conserve the move; otherwise, we reject it. The shift is first to the right, then to the left, when a right movement is no more possible. To avoid cycles during the second phase of the heuristic, two successive moves giving the same performance measure of the objective function are forbidden.

We tested two criteria for the objective function. Firstly, the objective function was to minimize the linear objective f_1. As shown in Table 7, it takes a lot of iterations to converge. Secondly, the objective function was to minimize the non-linear objective f_2. Because a slight variation in the sum of loads may improve the standard deviation, contrarily to the first criterion, the heuristic converges faster as shown in Table 7. For the example given in Table 1, the standard deviation is equal to 12.13. The first move will be from (Round 2, output 4) to (Round 2, output 5). This shift will reduce the standard deviation to 11.5. Table 3 and Table 4 give the solution at the end of the first and third phases.

Notations:

- $O_{\leftarrow}^{(r,k)}(s)$ is the set of direct left empty outputs of the non-empty output k in round r of solution s. For the example presented in Table 2, $O_{\leftarrow}^{(3,4)}(s) = \{3\}$, and $O_{\leftarrow}^{(4,4)}(s) = \emptyset$.
- $O_{\rightarrow}^{(r,k)}(s)$ is the set of direct right empty outputs of the non-empty output k in round r of solution s. For the example presented in Table 2, $O_{\rightarrow}^{(3,4)}(s) = \{5\}$ and $O_{\rightarrow}^{(3,2)}(s) = \{3\}$.

Algorithm 1 Procedure NEIGHBOR

Input: An initial solution $s^{(0)}$ and a tolerance value τ.
Output: A potentially new solution better than $s^{(0)}$.

1. Set $i := 0$ and $r := 1$.
2. Let k be the most loaded output among the non-empty ones of round r for solution $s^{(i)}$. If $O_{\rightarrow}^{(r,k)}(s^{(i)}) \neq \emptyset$, then choose the least loaded output $q \in O_{\rightarrow}^{(r,k)}(s^{(i)})$ and go to Step 4.
3. If $O_{\leftarrow}^{(r,k)}(s^{(i)}) \neq \emptyset$, then choose the least loaded output $q \in O_{\leftarrow}^{(r,k)}(s^{(i)})$, otherwise go to Step 5.
4. Let $s^{(T)}$ be a solution obtained from solution $s^{(i)}$ by shifting the mail batch from output k to output q for round r. If $f(s^{(T)}) < f(s^{(i)}) + \tau$, then move to a new current solution, *i.e.*, set $i := i + 1$ and $s^{(i)} := s^{(T)}$.
5. If $r < |R|$, then set $r := r + 1$ and go to Step 2. Otherwise, stop and return solution $s^{(i)}$.

Algorithm 2 Heuristic OPTI-MOVE

Input: A current solution s.
Output: The best known solution $s^{(B)}$.

- **Stage 1**.
  1.1. Set tolerance value $\tau := 0$ and $s^{(B)} := s$.
  1.2. Let s^* be a solution provided by the procedure NEIGHBOR(s, τ). If $f(s^*) < f(s)$, then reset $s := s^*$, set $s^{(B)} := s^*$ and repeat Step 1.2.
- **Stage 2**.
  2.1. Set tolerance value $\tau := 0.05 \cdot f(s)$ and decrement step $\Delta := 0.02$.
  2.2. Let s^* be a solution provided by the procedure NEIGHBOR(s, τ). If $f(s^*) < f(s^{(B)})$, then set $s^{(B)} := s^*$. If $f(s^*) < f(s)$, then reset $s := s^*$ and repeat Step 2.2.
  2.3. Decrease the tolerance value $\tau := \tau - \Delta$ and reset $s := s^*$. If $\tau > 0$, then go to Step 2.2.
- **Stage 3**.
  3.1. Set tolerance value $\tau := 0$.
  3.2. Let s^* be a solution provided by the procedure NEIGHBOR(s, τ). If $f(s^*) < f(s^{(B)})$, then set $s^{(B)} := s^*$. If $f(s^*) < f(s)$, then reset $s := s^*$ and repeat Step 3.2. Otherwise, stop and return the best found solution $s^{(B)}$.

Round	Out. 1	Out. 2	Out. 3	Out. 4	Out. 5
1	10	7	3	8	4
2		5	6	1	7
3		8	9	3	
4	11			4	3
$\sum$	21	20	18	16	16

Table 3. Solution at the end of Stage 1 with f_2=2.28.

Round	Out. 1	Out. 2	Out. 3	Out. 4	Out. 5
1	10	7	3	8	4
2		5	6	1	7
3			8	9	3
4	11	4			3
$\sum$	21	16	17	18	17

Table 4. Best solution obtained with f_2=1.92.

6. NUMERICAL EXPERIMENTS

This section reports experimental results on academic and real instances from La Poste. First, section 6.1 compares the heuristic with the MILP program on academic instances. Second, section 6.2 focuses on the heuristic with a comparison between the linear objective f_1 versus the non-linear objective f_2 on real instances. Finally, section 6.3 studies convergence and CPU time of the heuristic depending on the objective function f_1 or f_2.

The MILP was implemented in CPLEX 12.6, and the heuristic in Python. The experiments were performed on a processor Inter® Core™ i7-1165G7 at 2.80GHz.

6.1 MILP versus heuristic on academic instances

To test the MILP program, we randomly generated a set of instances of various sizes (rounds varying from 12 to 120, outputs from 20 to 90, for each DP a load from 10 to 60) with 30 instances per line in table 5. The heuristic with minimizing the linear objective f_1 was compared to the MILP program on the same instances. Table 5 reports the average on 30 instances of, successively, the initial value of f_1, the best value obtained, and the runtime in seconds for each method. The last column of table 5 reports the percentage of unsolved instances by the MILP program. For example, a percentage of 33.33% of failure means the MILP program could not solve 10 instances on 30. For instances with rows greater or equal to 75 and outputs greater or equal to 50, the MILP program cannot find a solution. The MILP program was able to optimally solve only instances with rows less or equal to 30 and outputs less or equal to 20, i.e., the first three lines of table 5. For larger instances, either the heuristic outperforms the MILP program or the MILP program cannot solve the instances. Moreover, the MILP program gives, on average, a solution in more than a minute, while the heuristic gives, on average, a solution in 0.5 seconds, with a maximum of 2.22 seconds.

Instances	Perf. measure: Mean(f_1)			Time (s)		%Fail
30 per line	Initial value	MILP	Heur. Min(f_1)	MILP	Heur. Min(f_1)	MILP
12x20	411.33	11.33	76.66	60.25	0.15	0
20x20	703.0	12.0	81.33	60.63	0.08	0
30x15	1154.3	9.66	80.66	59.09	0.15	0
30x30	1056.3	236.66	117.0	61.92	0.23	0
40x25	1394.0	182.66	107.7	61.71	0.26	0
40x40	1241.0	1235.8	116.0	64.08	0.33	13.33
50x40	1519.0	1422.1	146.0	62.72	0.18	3.33
60x50	1826.7	1876.0	157.0	63.00	0.21	33.33
75x50	2267.0	-	146.3	-	0.62	100
80x55	2425.0	-	167.3	-	0.61	100
100x60	2521.0	-	144.0	-	0.85	100
120x90	3040.0	-	130.0	-	2.22	100

Table 5. MILP versus heuristic on academic instances

6.2 Experiments on real instances

This section compares the performance of the heuristic with regards to the objective functions Min(f_1) or Min(f_2) on real instances of La Poste. Table 6 reports the set of eight real instances, with rows varying from 112 to 228, columns from 76 to 216 and for each DP, a load greater or equal to 10. Table 6 compares the performance of the heuristic following the two objective functions, i.e., minimizing f_1 or minimizing f_2, regarding each performance measure, i.e., f_1 and f_2. For each performance measure, Table 6 reports its initial value, the best solution with the heuristic when the objective function is to minimize f_1, the best solution when the objective function is to minimize f_2 and the percentage of improvement between the two objective functions. For example, for the first instance 112x76, regarding performance measure f_1, its initial value is 1870, minimizing f_1 gives a performance measure of 100 for f_1 while minimizing f_2 gives a performance measure of 40 for f_1, and the percentage of improvement from f_1 to f_2 is 150%. Finally, a performance measure for f_2 lower than the minimum load, i.e. 10, is very hard to obtain. However, when minimizing f_2, the heuristic gives, on average, a solution of 10.67, with a maximum of 15.4, which is a very good result.

Minimizing f_2 always gives better results, whatever the performance measure of interest, f_1 or f_2. On average, minimizing f_2 gives an improvement of 355.4% regarding the performance measure f_1 and an improvement of 256.4% regarding the performance measure f_2, with regard to the objective function minimizing f_1. As the non-linear objective f_2 is more sensitive to the moves, it improves the objective function more frequently, allowing better results in fewer iterations as explained in section 6.3.

6.3 Study of the convergence

This section studies the convergence of the heuristic concerning the objective functions, Min(f_1) and Min(f_2). On the same instances as in the previous section, Table 7 reports the number of iterations for each objective function: Min(f_1) or Min(f_2). As the second one gives better results with fewer iterations, the `CPU` time in seconds is provided only for the second objective. Not only Min(f_2) always gives better results, but it also always converges faster. On average, Min(f_2) needs 1748 fewer iterations than Min(f_1) to converge. Finally, on real instances, the heuristic always finds a solution in less than a minute, which was the constraint set by La Poste to use it in an industrial context. On average, the solution is found in 21.9 seconds.

Figure 1 illustrates the convergence of the heuristic concerning the performance measure f_2 for each phase of the heuristic. For the first instance 112x76, the heuristic's second and third phases do not improve the performance measure obtained at the end of the first phase, whereas the other instances are improved at each phase. Moreover, this graph shows that the first phase of the heuristic gives the most significant improvement for the performance measure. For example, for instance 208x202, the percentage of improvement at the end of the first phase is 3233% with regard to the initial value, whereas it is 80% between phase 1 and phase 2.

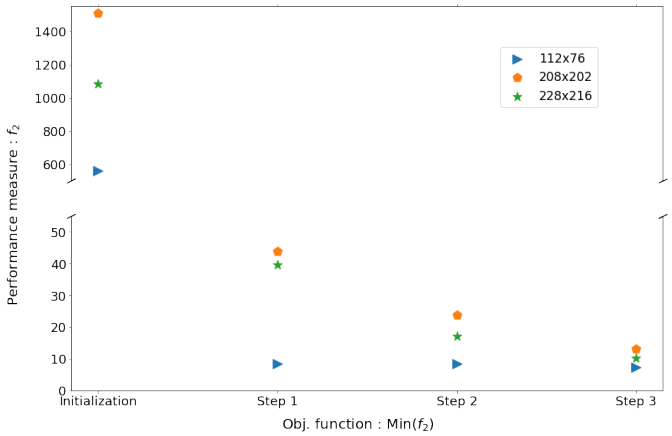

Fig. 1. The performance measure (f_2) depending on steps of the heuristic Min(f_2)

7. CONCLUSION

In this paper, we studied a variant of the assembly line balancing problem (ALBP), which consists of assigning tasks to an ordered sequence of stations so that the precedence relations among the tasks are satisfied and some performance measure is optimized. To the best of our knowledge, this paper presents the first application of ALBP to mail sorting, in which the outputs of the sorting machines are considered stations. The main difference consists of a particular structure of the mail precedence constraints, represented as a set of independent chains of the strict delivery order of the mail batches. An industrial case study at La Poste, the French postal company, is investigated.

We have provided a mathematical formulation of the problem to minimize a linear objective: the difference between the most and least charged outputs denoted f_1. To solve large instances provided by La Poste, we have proposed a new heuristic inspired by simulated annealing. It has been tested with two different objectives; firstly f_1, then a non-linear objective: the standard deviation of the outputs' loads denoted f_2. Experiments on real instances show that the heuristic performs well compared to the optimal solution. However, when the objective is to minimize f_2, it converges faster and gives better results, including the first criterion f_1. When minimizing f_2, the heuristic gives, on average, a solution of 10.67, with a maximum of 15.4, which is a very good result. Moreover, solutions for industrial instances are found in less than a minute. The heuristic has been successfully used in several industrial sorting centres of La Poste and is currently deployed across France.

REFERENCES

Azizoğlu, M. and İmat, S. (2018). Workload smoothing in simple assembly line balancing. *Computers & Operations Research*, 89, 51–57.

Battaïa, O. and Dolgui, A. (2013). A taxonomy of line balancing problems and their solution approaches. *International Journal of Production Economics*, 142(2), 259–277.

	Performance measure: f_1				Performance measure: f_2			
Instances	Initial	Obj. function: Min(f_1)	Obj. function: Min(f_2)	% Improvement between f_1 and f_2	Initial	Obj. function: Min(f_1)	Obj. function: Min(f_2)	% Improvement between f_1 and f_2
112x76	1870	100	40	150.0	623.9	19.1	8.3	129.2
111x217	2100	220	110	100.0	605.9	40.0	15.4	159.5
204x127	4110	170	140	21.4	1095.5	26.7	13.6	96.2
208x202	4450	200	100	100.0	1140.9	35.4	10.2	247.5
210x196	3850	230	20	1050.0	1161.5	43.4	7.1	512.0
213x214	4380	260	30	766.7	1313.3	49.0	9.7	404.0
226x202	4760	70	40	75.0	1206.3	22.1	8.2	168.6
228x216	5230	340	50	580.0	1573.3	55.3	13.0	334.1

Table 6. Comparisons of the objective functions Min(f_1) and Min(f_2) on performance measures f_1 and f_2

	#Iterations		Time (s)
Instances	Min(f_1)	Min(f_2)	Min(f_2)
112x76	245	45	0.28
111x217	1970	453	19.8
204x127	493	55	1.53
208x202	2097	356	23.7
210x196	2535	352	25.2
213x214	3201	418	36.1
226x202	2317	381	26.3
228x216	3627	437	42.1

Table 7. Convergence of the heuristic, w.r.t. the objective function

Boysen, N., Fliedner, M., and Scholl, A. (2007). A classification of assembly line balancing problems. *European Journal of Operational Research*, 183(2), 674–693.

Boysen, N., Schulze, P., and Scholl, A. (2022). Assembly line balancing: What happened in the last fifteen years? *European Journal of Operational Research*, 301(3), 797–814.

Eswaramoorthi, M., Kathiresan, G.R., Jayasudhan, T.J., Prasad, P.S.S., and Mohanrama, P.V. (2012). Flow index based line balancing: a tool to improve the leanness of assembly line design. *International Journal of Production Research*, 50(12), 3345–3358.

Ho, J.C., Tseng, T.L., Ruiz-Torres, A.J., and López, F.J. (2009). Minimizing the normalized sum of square for workload deviations on m parallel processors. *Computers & Industrial Engineering*, 56(1), 186–192.

Kim, Y.J., Kim, Y.K., and Cho, Y. (1998). A heuristic-based genetic algorithm for workload smoothing in assembly lines. *Computers & Operations Research*, 25(2), 99–111.

Kim, Y.K., Kim, Y.J., and Kim, Y. (1996). Genetic algorithms for assembly line balancing with various objectives. *Computers & Industrial Engineering*, 30(3), 397–409.

Nearchou, A.C. (2008). Multi-objective balancing of assembly lines by population heuristics. *International Journal of Production Research*, 46(8), 2275–2297.

Nearchou, A.C. (2011). Maximizing production rate and workload smoothing in assembly lines using particle swarm optimization. *International Journal of Production Economics*, 129(2), 242–250.

Otto, A. and Scholl, A. (2011). Incorporating ergonomic risks into assembly line balancing. *European Journal of Operational Research*, 212(2), 277–286.

Ouazene, Y., Yalaoui, F., Chehade, H., and Yalaoui, A. (2014). Workload balancing in identical parallel machine scheduling using a mathematical programming method. *International Journal of Computational Intelligence Systems*, 7($1^{\texttt{bis}}$), 58–67.

Pastor, R. (2011). LB-ALBP: the lexicographic bottleneck assembly line balancing problem. *International Journal of Production Research*, 49(8), 2425–2442.

Pinnoi, A. and Wilhelm, W.E. (1997). A branch and cut approach for workload smoothing on assembly lines. *INFORMS Journal on Computing*, 9(4), 335–350.

Rachamadugu, R. and Talbot, B. (1991). Improving the equality of workload assignments in assembly lines. *International Journal of Production Research*, 29(3), 619–633.