applied sciences

Article

Facility Layout Problem with Alternative Facility Variants

Jiří Kubalík *, Lukáš Kurilla † and Petr Kadera †

Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague, 16636 Prague, Czech Republic
* Correspondence: jiri.kubalik@cvut.cz
† These authors contributed equally to this work.

Abstract: The facility layout problem is one of the fundamental production system management problems. It has a significant impact on overall system efficiency. This paper introduces a new facility layout problem that allows for choosing from multiple variants of each facility. The need for choosing the most suitable selection from the facility variants while at the same time optimizing other layout quality indicators represents a new optimization challenge. We build on our previous work where single- and multi-objective evolutionary algorithms using indirect representation were proposed to solve the facility layout problem. Here, the evolutionary algorithms are adapted for the problem of facility variants, including the new solution representation and variation operators. Additionally, a cooling schedule, whose role is to control the exploration/exploitation ratio during the course of the optimization process, is proposed. It was inspired by the cooling schedule used in the simulated annealing technique. The extended evolutionary algorithms have been experimentally evaluated on two data sets, with and without the alternative variants of facilities. The obtained results demonstrate the capability of the extended evolutionary algorithms to solve the newly formulated facility layout problem efficiently. It also shows that the cooling schedule improves the convergence of the algorithms.

Keywords: facility layout problem; evolutionary algorithms; numerical modeling; multi-objective optimization; performance analysis

check for updates

Citation: Kubalík, J.; Kurilla, L.; Kadera, P. Facility Layout Problem with Alternative Facility Variants. *Appl. Sci.* **2023**, *13*, 5032. https://doi.org/10.3390/app13085032

Academic Editors: Guido Tosello, Roque Calvo and José A. Yaguë-Fabra

Received: 30 January 2023
Revised: 5 April 2023
Accepted: 12 April 2023
Published: 17 April 2023

1. Introduction

The facility layout problem (FLP) is one of the most important problems in production management and industrial engineering. It is a nonlinear combinatorial optimization problem defined as finding the most efficient arrangement of non-overlapping facilities on the factory floor with respect to one or more objectives subject to various constraints. Typically, the optimization goals are to maximize the utilization of the area available, minimize material-handling costs, or fulfill the adjacency or distance requirements between facilities. Furthermore, various constraints are imposed on the solution sought, such as the total area available, maximum acquisition or material handling costs, maximum maintenance costs, and other indirect expenses limits. The problem of finding the optimal layout of a set of elements occurs at various levels of a complex manufacturing environment, e.g., when arranging machines in a workshop, production lines in a production hall, or buildings on factory premises [1–3].

The FLP is a challenging task that has a direct impact on production efficiency, as it directly affects manufacturing costs, work in process, lead times, and productivity. A good layout of facilities contributes to the overall efficiency of operations and can reduce total operating costs by 20% to 50% [1]. Additionally, it has been shown that more than 35% of system efficiency is likely to be lost by applying incorrect layouts and location designs [4].

Numerous variants of the FLP problem have been formulated and researched. They can be categorized by different aspects as static or dynamic FLP, single- or multi-objective FLP [5], single- or multi-floor FLP, or FLP with equal or unequal facility areas, etc. [6–9].

When considering facility characteristics, several factors and design issues are addressed in the literature, such as the production variety and volume, the facility shapes and dimensions, the material handling system chosen, and different possible movement types allowed for parts, to name a few. Regarding the shapes and dimensions, there are two types of facilities: facilities with regular rectangular shapes and those with irregular shapes, i.e., generally polygons. In the former case, a facility can be defined by fixed length and width, and this facility is called fixed or rigid block [10]. An important FLP variant with numerous practical uses is the FLP with loosely-defined facility shapes and dimensions. In that case, each facility is associated with its area and a shape constraint, e.g., the maximum aspect ratio allowed (quotient between greater and lesser side measurements) or the minimum side length.

Layout problems are known to be generally NP-hard, which means that no exact algorithm exists that can provide an optimal solution in a reasonable polynomial time. On the one hand, there are many works in the literature that successfully use mathematical programming approaches to solve particular FLP variants. For instance, discrete quadratic assignment problem models were used for FLP with regular-shape equal-area facilities [11,12], and continuous linear and non-linear mixed integer programming models were used for FLP with irregularly-shaped facilities of unequal areas [13–17]. On the other hand, these approaches are applicable only to a rather small number of FLP instances, typically a few dozen facilities at most. Thus, many recent developments in solving the FLP are based on iterative metaheuristic approaches such as Tabu Search [18], Ant System [19,20], Particle Swarm Optimization [21], Variable Neighborhood Search [22], Simulated Annealing [23], and Evolutionary Algorithms [24–31].

We build on our previous work [32] that considers the FLP with workstations to be optimally placed into a production hall. A realistic definition of the hall and workstations was introduced there. The hall is defined as a rectangular area with obstacles (i.e., where no workstation can be placed) and communications (i.e., the areas that can be used only by relevant parts of the workstations). Workstations have a rectangular working area to which one or more handling zones can be attached. Furthermore, input/output points are optionally defined as well. These are used to define dependencies among workstations, such as links for material or semi-product handling. Each workstation can appear in the floor plan in one of six possible orientations. To solve this FLP, single- and multi-objective evolutionary algorithms have been proposed in [32]. The main feature of the evolutionary algorithms is that they use an indirect representation. This means that the exact position and rotation of the workstations are not directly encoded into the solution representation. Instead, the floor plan is represented indirectly using a so-called priority list that encodes the order and how the workstations will be added to the floor plan.

In this paper, we propose a new FLP variant with a new definition of loosely-defined facilities. Here, the facilities are not allowed to freely change their shapes within allowed boundaries. Instead, there are only several possible fixed-shape alternatives defined for each facility. The optimization task is to generate an optimal layout using exactly one variant per facility. Therefore, the optimization algorithm can not morph the shape of facilities arbitrarily. Instead, it has to choose among the feasible alternatives available for each facility. An important aspect of this FLP formulation is that the alternatives for a particular facility differ not only in shape and dimensions but also in other characteristics such as construction costs, operating costs, etc. Thus, a given selection of workstations' alternatives affects various quality and performance indicators of the corresponding layout.

This work was motivated by our research activities carried out in cooperation with the ŠKODA AUTO a.s. company. Our joint project aimed to propose machine learning-based approaches to optimally place multiple assembly lines' workstations within a production hall, given particular objectives and constraints. This is an optimization problem that occurs every time the product portfolio assigned to the respective hall is to be changed. Then, the assembly lines must be re-configured accordingly and placed in the production hall most effectively while respecting the products' assembly workflow, production hall infrastructure, logistics, and other requirements and constraints. An essential part of this optimization process is the re-design of some of the existing workstations and the design of new ones. The idea behind the possible use of multiple workstation variants is inspired by real-life best practices used by the experts when designing feasible layouts of the workstations. Usually, many constraints and restrictions affect the workstation's shape, such as internal dependencies related to the particular technology implemented by the workstation, the workstation's interface with the outside world, etc. Thus, only a few well-defined realizations might be considered for a given workstation.

To solve this FLP, we adopt the approach presented in [32] and further extend it so that multiple alternatives of each workstation can be defined and considered during the optimization process. To improve the exploration capabilities of the evolutionary algorithms, a new variation operator is defined for the extended representation. Additionally, a cooling schedule similar to the one used in simulated annealing [33,34] is introduced. Its role is to control the search mode along the optimization process. It slowly turns the search mode from a global search to a local one in order to restrict variation operators' scope to the tail section of the priority list as the algorithm converges. All in all, the primary goal of this paper is to introduce a new variant of FLP, its properties, and the implications for the solutions one can get, compared to the original FLP formulation with a single alternative defined for each workstation.

The main contributions of this work are:

- A new FLP with alternative variants of workstations was formulated. The use of the alternatives is a new FLP feature and represents a new challenge for optimization methods. To the best of our knowledge, this type of FLP has not yet been solved in the literature.
- Single- and multi-objective evolutionary algorithms to solve the problem were proposed.
- A cooling schedule that controls the search mode during the optimization process was incorporated into the evolutionary algorithms.
- The efficiency of the evolutionary algorithms and the impact of the cooling schedule on their performance at solving the new FLP problem were analyzed.
- Altogether, the new FLP formulation with redundant workstation definitions plus the proposed algorithms for finding high-quality plant layouts make up a truly flexible manufacturing system [35], which further contributes to sustainability within the whole production enterprise.
- A data set for the new FLP formulation (possibly a new benchmark in the field) was created. It will be made publicly available upon publication of this paper.

The paper is organized as follows. The problem solved in this work is defined in Section 2. Section 3 provides a concise description of the most important features of the approach we build on here. The proposed representation and algorithmic extensions that allow for using alternative workstation variants are described in Section 4. Section 5 describes the experiment set-up. Results obtained with the compared methods are presented and discussed in Sections 6 and 7. Finally, Section 8 concludes the paper.

2. Problem Definition

The FLP solved in this work is defined as follows. Givens include a particular specification of the hall, a set of N workstations $\mathcal{W} = \{w_1, \ldots, w_N\}$, and a set of U pairwise dependencies between workstations $\mathcal{L} = \{l_1, \ldots, l_U\}$. The hall specification defines its geometry, including restricted areas and internal communications; see Figure 1a. Each workstation is defined by its geometry, including the working area, handling areas, and position of its input/output points, and by other characteristics such as its size and cost. At least one of the workstations is defined in two possible variants, which can differ in any of the workstation's characteristics; see Figure 1b. The dependencies $\mathcal{L}$ define a set of source–destination links between workstations. Each link $l_i = (w_j, w_k)$ defines a directed connection between the output point of w_j and the input point of w_k. These are typically used to represent the material flow between workstations. The goal is to find a floor plan containing exactly one variant of each workstation that is optimal with respect to the given optimization objective(s) subject to the no-overlap constraint imposed on all workstation–workstation pairs as well as workstation–hall pairs. See [32] for formal definitions of the constraints.

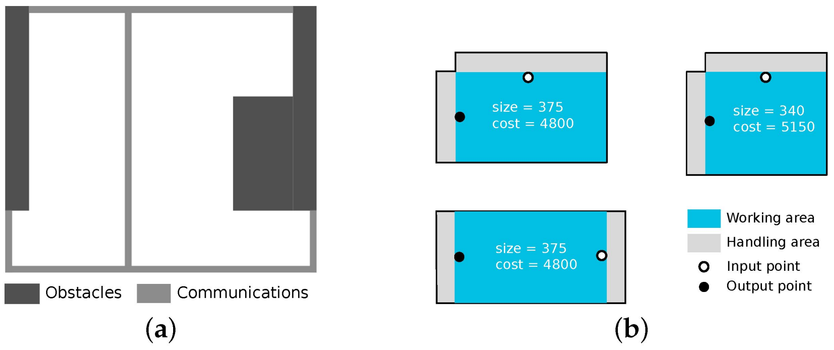

Figure 1. An example of (**a**) the production hall geometry and (**b**) three alternative variants of the same workstation.

In the following, we use the term *FLP with alternatives* for this new FLP formulation and the term *FLP without alternatives* for the original FLP as proposed in [32].

3. Preliminaries

The main feature of the original method introduced in [32] is that the evolutionary algorithms (see the pseudocode in Appendix A) use an indirect representation. This means that the layout of the floor plan, i.e., the exact position and rotation of the workstations, is not directly encoded into the solution representation. Instead, the solution is represented using a so-called *priority list*, which encodes the order in which the workstations will be added to the floor plan using placement heuristics designed for the problem. The priority list is thus a sequence of tuples

$$t_j = (\mathtt{id}, \mathtt{orientation}, \mathtt{heuristic}),$$

where each tuple at the position $j = 1 \dots N$ in the priority list specifies which workstation (`id`) with what orientation (`orientation`) will be added to the floor plan using what placement heuristic (`heuristic`). Mapping from the priority list to the floor plan is accomplished through an iterative process. It starts with an empty floor plan. Then, the priority list is traversed from left to right, and the workstations are successively added one by one to the developed floor plan according to the 'instructions' given in the corresponding tuple. The mapping process is illustrated in Figure 2. For more details, refer to [32].

An important aspect of this representation is its redundancy. On the one hand, each priority list maps to exactly one floor plan. On the other hand, multiple different priority lists can potentially map to the same floor plan. This is an important feature, as it allows for converging to an optimal solution via different search trajectories.

In [32], a single crossover and several mutation operators were used. The crossover operator is a variant of a standard order-based crossover operator [36]. It takes for its input two parental priority lists and starts generating the offspring by randomly choosing two crossover points. Standard tournament selection is used to select the parents. Then, the head and tail parts are inherited from the first parent. The middle part is filled in with the missing elements in the same order in which they appear in the second parent. The crossover points delimiting the head, middle, and tail parts are found randomly anew for each crossover operation. In the following, we denote this operator as a *2-point crossover*. We also implement a simple *1-point crossover* operator. It generates the offspring by inheriting a randomly-chosen head part from the first parent and then appending missing elements in the same order in which they appear in the second parent.

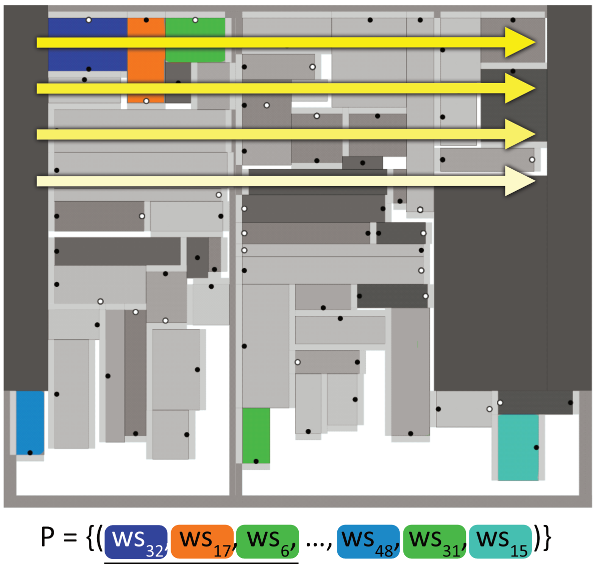

Figure 2. Illustration of the effect of mapping of the priority list `P` to the floor plan. Each term $\mathtt{ws}_\mathtt{i}$ stands for a tuple representing a particular variant of the workstation *i*. Workstations are placed into the floor plan in a top-down manner as indicated by the yellow arrows of decreasing intensity. The head part of the priority list ($\mathtt{ws}_{32}$, $\mathtt{ws}_{17}$, and $\mathtt{ws}_{6}$) maps to the top region of the floor plan. The tail part of the priority list ($\mathtt{ws}_{48}$, $\mathtt{ws}_{31}$, and $\mathtt{ws}_{15}$) maps to the bottom region of the floor plan.

The mutation operators operate on a single priority list. They are designed to modify a randomly-chosen element of the parental priority list so that it changes either its position within the priority list or the way the corresponding workstation is added to the floor plan, i.e., its `orientation` or `heuristic`.

In the original work, two optimization criteria, `maxFreeSpace` and `minConnDist`, were defined. The first one maximizes the compactness of the floor plan, defined as the maximization of the free space remaining between the lower horizon of the placed workstations and the bottom edge of the hall. This objective ensures that among the maximally-compact floor plans, the ones with the remaining space concentrated towards the bottom edge of the hall are preferred. As its name suggests, this objective drives the optimization towards floor plans that have the greatest potential for placing new workstations. This objective exhibits desired properties, as it drives the search towards maximally-compact floor plans and simultaneously maximizes the utilization of the available area. Contrary to traditional variants of FLP space occupation-related objectives that try to pack the entities within the smallest rectangular envelope, our `maxFreeSpace` objective better optimizes for space utilization. The `minConnDist` objective minimizes the total length of all connections between workstations calculated as the sum of Euclidean distances between the output point of the source workstation and the input point of the destination workstation over all pairs of interconnected workstations in $\mathcal{L}$; see [32] for details.

4. Proposed Method

4.1. Extended Representation

Here, we extend the representation so that it allows for multiple variants of each workstation. The priority list is composed of tuples $t_i^{'}$ of the following form

$$t_j^{'} = (\mathtt{id}, \mathtt{orientation}, \mathtt{heuristic}, \mathtt{variant}),$$

with the new attribute `variant` defined as a tuple (`geometry`, `size`, `cost`), where

- `geometry` is a definition of the workstation's variant geometry, including its shape, position of its handling areas, and the position of its input/output points (see Figure 1b),
- `size` is the size of the working area of this variant, and

- `cost` is the cost of this particular realization of the workstation.

There are n_i alternative variants defined for each workstation w_i. Thus, the size of the priority list is $M = \sum_{i=1}^{N} n_i$. The whole solution, i.e., the floor plan, is represented by an extended priority list, which is a permutation of a set of tuples T that contains all variants of all workstations

$$T = \{t'_{1,1}, \ldots, t'_{1,n_1}, \ldots, t'_{N,1}, \ldots, t'_{N,n_N}\}. \qquad (1)$$

The extended priority list is interpreted via an iterative process similar to the one used for the original single-shape representation version. The priority list is traversed from left to right, and the corresponding workstations are added to the floor plan in a first-come-first-served manner. This means that the left-most occurrence of each workstation's variant within the priority list is used in the floor plan; i.e., it is an active variant. The remaining inactive ones are just ignored.

4.2. Mutation

In order to reflect the extended representation and the fact that the search space significantly increases with the introduced workstation alternatives, we have added a new mutation operator to the evolutionary algorithms. This operator swaps two elements of the priority list associated with the same workstation. Given the parental priority list P as a permutation of set T, see (1), the mutation works in the following steps:

1. Randomly choose a position k within the parental priority list P;
2. Find a set of positions S_k in P, excluding the position k, that are associated with the same workstation as the element $P[k]$,
   $S_k = \{l : (P[l].\texttt{id} = P[k].\texttt{id}) \wedge (l \neq k)\}$;
3. Randomly choose one position j from S_k;
4. Swap elements at positions j and k.

If one of those two swapped elements is the left-most occurrence of the workstation in the priority list, then this action most likely leads to the effective modification of the corresponding floor plan. Otherwise, the mutation operation does not affect the floor plan represented by the priority list. This is a so-called silent mutation, and it has been shown to be beneficial for preserving diversity within the population in evolutionary algorithms [37,38].

Note that each workstation variant can be present in the priority list with only 1 out of 12 configurations, given there are 6 possible orientations and 2 placement heuristics. Variants present in the priority list can change their configuration via standard mutations that change the `orientation` and `heuristic`. Importantly, this new mutation operator works with particular workstation variant configurations, as they are stored in the priority list. By executing the effective mutation, a new variant of a certain workstation with its particular configuration becomes active (i.e., will be used in the floor plan). However, the other one, which becomes inactive, stays intact with its configuration in the priority list, thus available for its later re-use.

4.3. Cooling Schedule

Simulated annealing (SA) is a metaheuristic optimization technique. It searches for the optimal solution in an iterative process, where in each iteration, it generates a random neighbor of the current solution and either accepts the neighbor as the new current solution or rejects it. The neighbor is always accepted if it is not worse than the current solution. It may also be accepted with probability p^c, even if it is worse than the current solution. The probability p^c is high at the beginning of the run and gradually decreases through time, approaching zero towards the end of the run. This way, SA gradually changes its search mode from a global exploration to a local one. The probability p^c changes according to the *cooling schedule*, which is either defined by the user or automatically adapted during the course of the run [34].

Here, we adopt a similar approach to control the search mode along the optimization process. The idea behind it is based on the relation between the priority list and the mapping process. The mapping process starts at the upper-left corner of the empty hall. Then, the workstations are added to the floor plan one by one, filling up the available hall space in a left-right and top-down manner. The workstations are served in the order they come in the priority list; see Figure 2. This means that changes made to the head part of the priority list have a greater impact on the resulting floor plan than changes made within the tail part of the priority list. This is reflected by the cooling schedule proposed in this work, as it gradually restricts variation operators' scope to the tail section of the priority list as the algorithm converges. It is realized by preventing the variation operators (i.e., crossover and mutations) from perturbing elements of the priority list whose position is less than or equal to the left-most variation point. In this way, the search mode slowly turns from a global one to a local one.

There are many possible ways to define the cooling schedule. The one used in this work is shown in Figure 3. It sets the left-most variation point to 1 for the first half of the run. Then, it jumps to $|T|/4$, making the leading quarter of the priority list fixed. As the run progresses towards the end, the fixed portion of the priority list linearly increases to $3 * |T|/4$. Thus, in the first half of the run, the algorithm works in a standard global exploration mode. Then, the operation mode changes to the local one. However, even in the final phase of the run, the exploration capabilities stay at a significant level, as the tail quarter of the priority list can still be modified by variation operators.

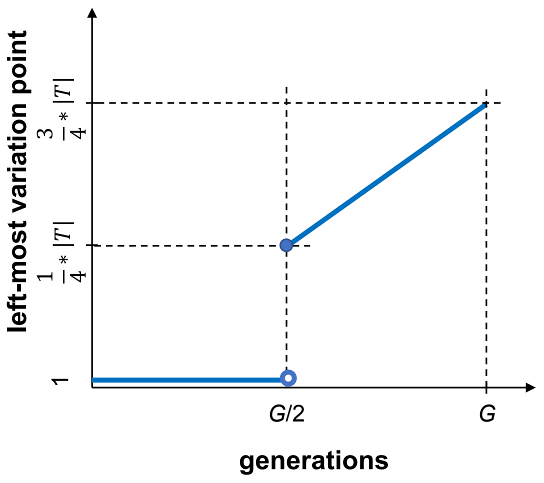

Figure 3. Cooling schedule.

4.4. Optimization Criteria

In this work, we use two optimization criteria defined in [32]—the `maxFreeSpace` and `minConnDist`. The `maxFreeSpace` objective maximizes the free space concentrated towards one side of the hall. The `minConnDist` objective minimizes the sum of Euclidean distances between input and output points of interconnected workstations.

In addition to the two optimization criteria, the `minCost` objective is defined to make use of the newly defined `cost` attribute of the workstation. It minimizes the sum of `cost` values of particular workstation variants used in the given floor plan. Note that this is an optimization objective that explicitly defines quality for any particular selection of the workstations' variants. In the following, we use the term *explicit objective* in connection with the `minCost`. On the contrary, the `maxFreeSpace` and `minConnDist` objectives are *implicit* ones in the sense that they are not directly calculated out of the parameter values of the selected workstations' variants. Instead, their value depends on how well the workstations' variants are placed in the floor plan.

The `size` and `cost` attributes were added to the workstation definition in order to illustrate another possible dimension of the optimization problem. With these new attributes, the optimization task can be formulated as a multi-objective optimization problem that seeks trade-off solutions to optimize both the implicit objective, i.e., `maxFreeSpace`, and the explicit objective, `minCost`.

5. Experiments

In this section, the experimental setup is described. First, the data sets are described. Then, the compared algorithms are listed together with their configurations used in the experiments. Finally, the experiments, tested hypotheses, and the evaluation scheme used to analyze the algorithms' performance are described.

5.1. Data Set

Two data sets are used in the experiment:

- Data set D_A (data set without alternative workstation variants)—a data set derived from the one used in [32]. It contains 58 workstations, each provided in just a single variant. The length of the priority list is $|T| = 58$. New attributes `cost` and `size` have been added to the workstation definitions. The total cost of all workstations is $cost_{static} = 284.200$. The total working area of the workstations is 33.829 m^2.
- Data set D_B (data set with alternative workstation variants)—a data set derived from the data set D_A so that a second variant has been defined for 30 randomly-chosen workstations out of the original 58. Thus, the length of the priority list increases to $|T| = 88$. The two variants for each such workstation differ in both the working area size and cost. The size is inversely proportional to the cost. The total cost of the set of workstations with only the lower-cost variants and the set of workstations with only the higher-cost variants is 238,900 and 329,000, respectively.

5.2. Compared Algorithms

Single-objective (EA) and multi-objective evolutionary algorithms (MOEA) are used in the experiments. We use the notation 'A_V_C' to refer to the particular algorithm's variant, with $A \in$ {EA, MOEA}, $V \in$ {T, F} denoting whether the alternative workstations' variants are used (T—true) or not (F—false) and $C \in$ {T, F} denoting whether the cooling scheme is used or not. The evolutionary algorithms were run with the following configuration:

- Population size: 200;
- Maximum number of generations (G): 1000;
- Tournament size: 4;
- Probability of crossover: 0.6;
- Probability of mutation: 0.5.

Note that the algorithms were run with the same parameter setting in all experiments. No parameter tuning was carried out, since one of the goals of the experiments is to demonstrate the viability and robustness of the proposed algorithms, given that no knowledge about the best values of the parameters is available.

5.3. Goals of Empirical Investigation

The following three experiments were carried out to evaluate the performance of the single- and multi-objective evolutionary algorithms on the FLP with alternatives:

- Experiment 1—The goal of this experiment was to analyze and compare the performance of the algorithms on the FLP with and without alternatives. Note that the search space in the case of the FLP with alternative shapes is much bigger than the search space of the single-shape FLP. Thus, the question was whether the algorithms could converge to solutions that were at least as good as the solutions found on the single-shape FLP. Experiments with both the EA and MOEA were carried out. In the former case, the `maxFreeSpace` optimization objective was used. In the latter one, the `maxFreeSpace` and `minConnDist` objectives were used. Note that, in this case, both optimization objectives were of the implicit type, as the workstation cost was not considered here.

- Experiment 2—The goal of this experiment was to investigate the effect of using the cooling scheme on the quality of solutions. Again, experiments with the EA and MOEA were carried out while optimizing the same objectives as in Experiment 1.
- Experiment 3—The goal of this experiment was to investigate the performance of the MOEA, given the optimization objectives were a mixture of explicit and implicit ones. The MOEA was run with the implicit `maxFreeSpace` and the explicit `minCost` objectives. As mentioned in Section 4.4, the `minCost` objective can easily be improved just by choosing the cheaper variant for each workstation. On the contrary, improving the `maxFreeSpace` objective requires finding a better placement for the given selection of the workstation variants. So, it is likely that the algorithm may prematurely converge to the set of solutions composed of low-cost/high-size workstation variants, thus exhibiting low `maxFreeSpace` values. Thus, the question was whether the MOEA could efficiently search for a diverse set of high-quality solutions.

5.4. Evaluation

One hundred independent runs were carried out with each tested algorithm's variant in each experiment. In the case of the single-objective optimization, the median best-of-run values were compared. In the case of multi-objective optimization, there is typically no single best solution produced by the MOEA. Instead, a set of so-called non-dominated solutions is delivered by the MOEA at the end of its run. We used a so-called *hypervolume* performance metric [39], frequently used in the literature, to assess the quality of the set of non-dominated solutions. It calculates the hypervolume of the multi-dimensional region enclosed by the set of non-dominated solutions and a reference point; see Figure 4.

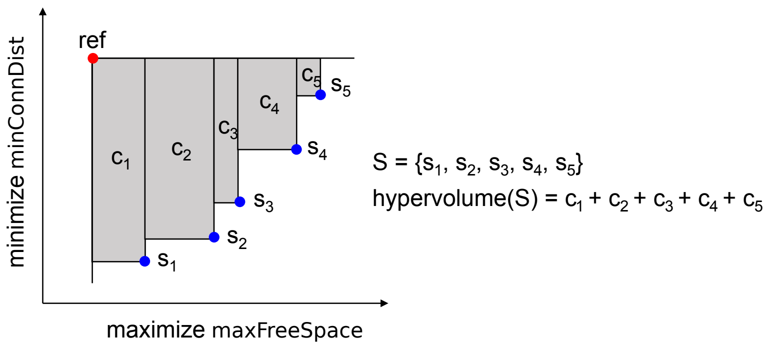

Figure 4. Illustration of the hypervolume calculation for a set of non-dominated solutions S, given the maximization objective `maxFreeSpace` and minimization objective `minConnDist`.

The coordinates of the reference point were calculated as the worst value of the respective objective observed in the final non-dominated fronts over all 100 runs. The hypervolume expresses the size of the region that is dominated by the non-dominated set. Similarly to the single-objective case, we then compared the median of the hypervolume values.

Note that the same random generator seeds were used for all series of 100 independent runs in all experiments. This implies that the runs with the cooling schedule followed the same search trajectory as those without the cooling schedule, up to generation 500 (i.e., $G/2$). Then, the two trajectories departed from each other and went their own way. This allowed us to accurately measure the effect of the cooling schedule.

In order to assess the statistical significance of the differences among the methods, we analyzed them pair-wise using the Wilcoxon rank-sum test, which rejects the null hypothesis that the compared result sets are sampled from continuous distributions with equal medians at the 5% significance level.

6. Results

6.1. Experiment 1

First, we analyzed the performance of the evolutionary algorithms on the extended FLP with alternative workstations' variants. Tables 1 and 2 show the results obtained for the single-objective and multi-objective cases, respectively. In Table 1, the best median `maxFreeSpace` value of 10,501 was observed for the variant EA_F_T (i.e., without alternatives, with cooling). This is significantly better than the median `maxFreeSpace` value achieved by EA_T_T. Also, the results obtained by EA_F_F are slightly better than those achieved by EA_T_F, although not significantly so as indicated by the p-value of 0.37.

Table 1. Results for the single-objective case of Experiment 1 (variants = false vs. variants = true) and Experiment 2 (cooling = false vs. cooling = true). The median of the set of 100 best-of-run values is presented for each method plus the p-values calculated pairwise with the Wilcoxon rank-sum test. The statistically significant differences are highlighted in bold.

	Median	***p*-Value**			
	`maxFreeSpace` **[m^2]**	**EA_T_T**	**EA_T_F**	**EA_F_T**	**EA_F_F**
EA_T_T	10,176	-	$\mathbf{2.2 \times 10^{-8}}$	$\mathbf{3.1 \times 10^{-2}}$	$\mathbf{1.6 \times 10^{-6}}$
EA_T_F	9879	-	-	$\mathbf{2.9 \times 10^{-13}}$	0.37
EA_F_T	10,501	-	-	-	$\mathbf{1.6 \times 10^{-11}}$
EA_F_F	9940	-	-	-	-

Table 2. Results for the multi-objective case of Experiment 1 (variants = false vs. variants = true) and Experiment 2 (cooling = false vs. cooling = true). The median of the set of 100 hypervolume values is presented for each method plus the p-values calculated pairwise with the Wilcoxon rank-sum test. The best values are highlighted in bold.

	Median	***p*-Value**			
	Hypervolume	**MOEA_T_T**	**MOEA_T_F**	**MOEA_F_T**	**MOEA_F_F**
MOEA_T_T	1.61×10^7	-	0.16	$\mathbf{9.76 \times 10^{-4}}$	7.97×10^{-2}
MOEA_T_F	1.55×10^7	-	-	$\mathbf{1.2 \times 10^{-5}}$	$\mathbf{2.13 \times 10^{-3}}$
MOEA_F_T	1.71×10^7	-	-	-	5.96×10^{-2}
MOEA_F_F	1.65×10^7	-	-	-	-

The same trend is also observed for the multi-objective optimization case; see Table 2. Again, the runs not using alternative variants yielded significantly better results than the corresponding runs with alternatives. Here, the difference between results obtained with and without alternatives was even stronger than in the single-objective case.

These observations confirm our hypothesis that the FLP with alternatives is significantly harder than the one without alternatives. The algorithms were not able to converge on the extended data set D_B to produce solutions of the same quality as the ones obtained from the data set D_A, given the same computational resources.

6.2. Experiment 2

In Table 1, a clear trend that the cooling schedule helps to converge to better solutions can be observed. EA_T_T significantly outperformed EA_T_F, and the same held for EA_F_T and EA_F_F.

Using the cooling schedule led to better solutions in the multi-objective optimization case as well, but the differences were not as profound as in the single-objective case. While MOEA_T_T is not significantly better than MOEA_T_F in terms of the median hypervolume, as indicated by the p-value of 0.16, the run using the cooling schedule resulted in a better set of non-dominated solutions compared to the corresponding run not using the cooling schedule, in 70 out of 100 runs. A similar ratio, 73:27, was observed for MOEA_F_T and MOEA_F_F. This indicates that the cooling schedule has a positive effect on performance, even in the multi-objective optimization scenario.

The observed difference in the effect of the cooling schedule in the single- and multi-objective optimization may also point to the fact that in the multi-objective case, it is even more important to keep the population maximally diverse throughout the whole run. This applies particularly to the head parts of individuals' priority lists. As soon as the cooling schedule becomes effective, the head parts get 'frozen' and cannot be modified any more. Suppose a majority of head parts of the priority lists in the population determine floor plans that are good in terms of one particular objective. In that case, there is only a small chance that additional floor plans, good in terms of the other objective, will effectively evolve from then on. In the single-objective case, this is not an issue.

6.3. Experiment 3

Figure 5 clearly demonstrates the ability of the MOEA to efficiently evolve a population of candidate solutions and to converge to a diverse set of high-quality solutions even when the optimization criteria are a mixture of explicit (`minCost`) and implicit (`maxFreeSpace`) ones. It shows all non-dominated solutions from the initial and final populations of all independent runs. One can see that, starting from very poor solutions with the initial population, the MOEA converges to much better solutions in the end. In particular, the final non-dominated fronts have shifted both to the right and downwards, thus significantly improving both objectives. However, the median `maxFreeSpace` value of the final non-dominated solutions still remains well below the `maxFreeSpace` value obtained with a single-objective EA from the data set D_A (i.e., the value of EA_F_T in Table 1). We hypothesize this can be attributed to the MOEA having to deal with a mixture of explicit and implicit objectives, where the explicit ones are much easier to improve than the implicit ones. Thus, the optimization algorithms need to be further enhanced to cope with this situation efficiently. On the contrary, the vast majority of final solutions lie below the $cost_{static}$ value of `minCost`. Many of them are also very good in terms of the `maxFreeSpace` value.

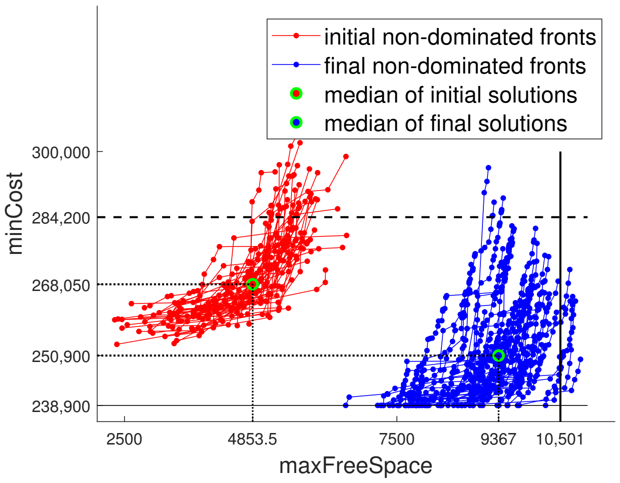

Figure 5. Evolution of non-dominated solutions using MOEA_T_T. Initial (red) and final (blue) sets of non-dominated solutions generated in 100 independent runs are shown. The horizontal dashed line indicates the `minCost` = $cost_{static}$ value of solutions produced for the data set D_A (i.e., without alternative workstations). The vertical solid line indicates the median `maxFreeSpace` value obtained with single-objective EA on data set D_A.

We also experimented with different genetic operators, namely the 1-point and 2-point crossovers. Figure 6 shows the final sets of non-dominated solutions obtained with MOEA_T_T using the 1-point and MOEA_T_T using the 2-point crossover. As for the median `minCost` value, the results are almost equal. A bigger difference, in favor of the 2-point crossover, can be observed with respect to the median `maxFreeSpace` value. The 2-point crossover also outperforms the 1-point one in terms of the hypervolume dominated by the final set of solutions. The median hypervolume dominated by the 2-point crossover solutions is 2.83×10^9, while the median hypervolume of the 1-point crossover solutions is 2.72×10^9. This is a statistically significant difference, as indicated by the p-value of 4.3×10^{-4}. This observation clearly shows the importance of using appropriate sampling operators.

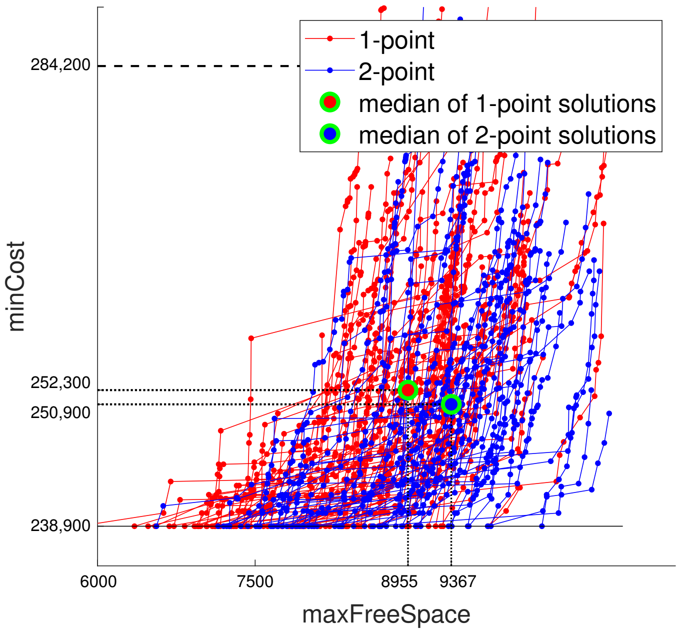

Figure 6. Comparison of MOEA_T_T using 1-point and MOEA_T_T using 2-point crossover. The sets of non-dominated solutions generated in 100 independent runs are shown.

Figure 7 shows illustrative examples of two floor plans evolved in a single run of MOEA_T_T. Several workstations are realized differently in these two floor plans; examples are the three workstations marked 'A', 'B', and 'C'.

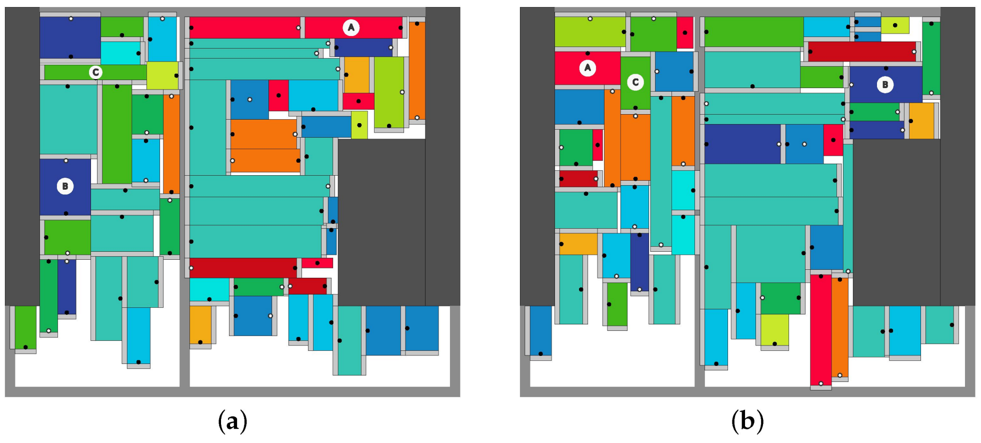

Figure 7. Examples of two floor plans using different workstation variants: (**a**) a floor plan with `maxFreeSpace` = 10.010 m^2 and `minCost` = 261.500. (**b**) a floor plan with `maxFreeSpace` = 10.620 m^2 and `minCost` = 243.900. Workstations marked with 'A', 'B', and 'C' are examples of workstations realized differently in the two floor plans.

7. Discussion

7.1. Applicability of the Proposed Approach

The proposed FLP, with the realistic definition of workstations together with the finite set of possible fixed-shape realizations available for each workstation, fills the current gap in the literature. Existing FLP formulations involving loosely-defined facility shapes and dimensions rely on the concept that the facilities can freely vary their shape and dimensions within some limits. However, this is not a general case. In many practical situations, facilities can only be realized by a small number of fixed-shape variants. One example is the automotive industry, with assembly lines composed of dozens of interconnected workstations. Every time a new car model is to be introduced to production, the workstations and assembly lines must be re-configured and reorganized within a production hall accordingly. Importantly, the design of a feasible workstation realization is a highly-constrained optimization problem itself. It involves various constraints related to the specifics of the technological procedure implemented by the workstation and to the workstation's interface with other workstations it interacts with.

7.2. Advantages and Disadvantages of (MO)EA

In general, EAs are optimization techniques with many advantages—they can be used for an arbitrary type of optimization problem (continuous, discrete, multimodal, single/multi-objective, nonlinear, constrained, etc.). Thus, they are particularly well-suited for the FLP problem as well.

From a practical point of view, EAs are easy to implement. They also allow easy incorporation of arbitrary constraints and optimization objectives. This is often much easier than extending the model for a given mathematical programming method.

Perhaps the most important advantage of the proposed approach is its ability to deliver a diverse pool of high-quality solutions. This is demonstrated in Figure 7, which illustrates the ability of the multi-objective algorithm to evolve high-quality non-dominated solutions in a single run while different workstation variants are automatically chosen for different floor plans. In general, this technique generates multiple solutions from which the user can choose the best one according to his/her expert knowledge. We believe that this is the feature that makes the method truly usable in practice. Often, when solving real-world problems, not all aspects of the desired solution can be captured within the formal definition of the optimization problem. Thus, only the most important ones and the ones that can be formally described are used to control the search process. Even then, it is very beneficial if the user is provided with a set of diverse, high-quality solutions from which he/she can further select the best-looking one in the end.

EAs also have a known limitation that they might be computationally intensive because they require processing the entire population of candidate solutions in each generation. On the other hand, when solving many real-world problems, including the FLP, the solver has enough time to generate a high-quality solution. Moreover, the EA can be interrupted at any time and return the current best solution generated thus far.

7.3. Potential for Improvement

While the experimental results show that the proposed evolutionary algorithms are effective and robust, there is still some room for further improvement. As mentioned above, it is absolutely essential that the evolutionary algorithm maintain a diverse set of candidate solutions during the course of the whole run while converging to ever-improving solutions. To attain this, the trade-off between exploration and exploitation has to be ensured. To achieve this, the algorithms have to ensure a balance between exploration and exploitation.

It is also important that efficient genetic operators are used to sample new candidate solutions. It was demonstrated that the performance of the 1-point crossover and the 2-point one differ significantly. This suggests that new types of operators for the indirect representation are worth exploring. Last but not least, the experiments showed that the multi-objective algorithm experiences difficulties when simultaneously optimizing explicit and implicit objectives. The algorithm tends to prioritize the explicit objective over the implicit one. This is not unexpected, as improving the explicit objective value is much easier than improving the implicit one. Techniques for diversity preservation such as novelty search, niching, or fitness sharing might help to remedy these issues.

8. Conclusions

We formulated a new FLP problem that involves alternative variants of facilities. This is a new FLP feature and represents a new challenge for optimization methods. The realistic definition of workstations and their variants suits many practical situations in factory floor optimization and management.

We proposed extensions to the single- and multi-objective evolutionary algorithms to solve the problem. Besides including a new mutation operator, a cooling schedule was proposed to control the search mode during the optimization process. It progressively restricts the scope of the crossover and mutation operators, thus changing the optimization mode from a global one at the early stages of the run to a local one at the end of the run. Experiments showed that the cooling schedule significantly improves the performance of the algorithms.

The efficiency of the extended evolutionary algorithms has been tested and analyzed for the data sets with and without alternative facility variants. It was demonstrated that the algorithms are able to efficiently solve the extended FLP, which involves a much larger search space than the FLP without alternatives. However, there is still room left for improvement of the optimization algorithms.

Finally, the performance of the multi-objective algorithm was analyzed within an optimization scenario that involves both explicit and implicit objectives, where the explicit ones are much easier to improve than the implicit ones. Although the algorithm demonstrated its potential to generate well-fit solutions, it also showed that its limits could be pushed even further. This remains a challenge for our future research.

In our future research, we will investigate different variants of the cooling schedule procedure. This can involve investigation of the linear vs. exponential schedule, static vs. dynamic setting of the schedule parameters, deterministic vs. probabilistic schedule, automatic identification of the trigger points based on the statistics of the current population, etc.

While the current formulation of the proposed FLP assumes only rectangular workstation shapes, it can be straightforwardly adapted to consider free-form, possibly non-convex, rectilinear shapes as well. New constraints and objectives can easily be introduced as well. This is also the topic of our further research.

Author Contributions: Conceptualization, J.K., L.K. and P.K.; Methodology, J.K., L.K. and P.K.; Software, J.K.; Validation, J.K.; Formal analysis, J.K.; Investigation, J.K., L.K. and P.K.; Data curation, J.K.; Writing—original draft, J.K., L.K. and P.K.; Funding acquisition, P.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the Ministry of Education, Youth, and Sport of the Czech Republic within the project Cluster 4.0 number CZ.02.1.01/0.0/0.0/16_026/0008432. The publication was created with support from the project "Regeneration of Used Batteries from Electric Vehicles" (Slovak ITMS2014+code 313012BUN5), which is part of the Important Project of Common European Interest (IPCEI) called European Battery Innovation (code OPII-MH/DP/2021/9.5-34), announced as a part of Operational Program Integrated Infrastructure.

Data Availability Statement: The data are available from the corresponding author upon request.

Acknowledgments: This work was funded by the Ministry of Education, Youth, and Sport of the Czech Republic within the project Cluster 4.0 number CZ.02.1.01/0.0/0.0/16_026/0008432. The publication was created with support from the project "Regeneration of Used Batteries from Electric Vehicles" (Slovak ITMS2014+code 313012BUN5), which is part of the Important Project of Common European Interest (IPCEI) called European Battery Innovation (code OPII-MH/DP/2021/9.5-34), announced as a part of Operational Program Integrated Infrastructure.

Conflicts of Interest: The authors declare no conflict of interest.

Nomenclature

The following nomenclature and abbreviations are used in this manuscript:

EA	Evolutionary Algorithm
FLP	Facility Layout Problem
FLP without alternatives	FLP with a single variant for each workstation
FLP with alternatives	FLP with multiple possible variants defined for each workstation
MOEA	Multi-Objective Evolutionary Algorithm
SA	Simulated Annealing
A	Flag indicating the evolutionary algorithm type, $A \in \{\text{EA, MOEA}\}$
C	Flag indicating the usage of the cooling scheme
$cost_{static}$	Total cost of the workstations of the FLP without alternatives
G	Maximum number of generations
$\mathcal{L}$	Set of pairwise dependencies between workstations
M	Size of the priority list for the FLP with alternatives
N	Number of workstations
n_i	Number of variants of the workstation w_i
p^c	Probability of accepting a worsening step in the SA method
P	Priority list
S_k	Set of positions in P, excluding the position k, that are associated with the same workstation as the element $P[k]$
t_j	Tuple representing the j-th element of the priority list, used with the FLP without alternatives
$t_i^{'}$	Extended version of t_i used with the FLP with alternatives
T	Set of tuples that contains all variants of all workstations
U	Number of pairwise dependencies between workstations
V	Flag indicating the FLP variant
w_i	i-th workstation
$\mathcal{W}$	Set of all workstations to be placed in the floor plan

Appendix A. Pseudocode of the Evolutionary Algorithms

Algorithm A1: Single-objective evolutionary algorithm, EA

```
Input: $PopSize$ ... population size
       $G$ ... maximum number of generations
       $P_C$ ... crossover rate
       $P_M$ ... mutation rate
       $EliteSize$ ... number of elite individuals
Output: Floorplan represented by the best individual

init($population$)
evaluate($population$)                                   # NSGA-II specific
$g \leftarrow 0$
while $g < G$ do
    $tempPop \leftarrow \{\}$
    while $|tempPop| < PopSize$ do
        $par1 \leftarrow$ select($population$)                # NSGA-II specific
        if rand() $< P_C$ then
            $par2 \leftarrow$ select($population$)            # NSGA-II specific
            $child \leftarrow$ crossover($par1, par2$)
            if rand() $< P_M$ then
                mutate($child$)
        else
            $child \leftarrow par1$
            mutate($child$)
        evaluate($child$)                                  # NSGA-II specific
        $tempPop \leftarrow tempPop + child$
    $tempPop \leftarrow tempPop \cup$ elite($population, EliteSize$)   # NSGA-II specific
    $population \leftarrow$ getUnique($tempPop, PopSize$)     # NSGA-II specific
    $g \leftarrow g + 1$
return bestOf($population$)                                # NSGA-II specific
```

The EA starts with a random initialization and evaluation of the initial population of individuals. Evaluation of an individual means that, first, the individual's priority list is mapped to the actual floor plan, and then the quality measure of the floor plan is calculated. Then, the algorithm iterates through G generations, lines 4–20. In each generation, a new population is created as follows. A population $tempPop$ is created through a standard evolutionary process using selection, crossover, and mutation. When the $tempPop$ of size $PopSize$ has been generated, a number of $EliteSize$ top best unique individuals from the previous generation are added to $tempPop$, line 18. Finally, the best $PopSize$ unique individuals from $tempPop$ are selected to the new $population$, line 19. In the end, the best floor plan of the last population is returned.

Multi-objective evolutionary algorithm, MOEA, implements the NSGA-II algorithm [40]. The pseudocode of the EA algorithm applies to MOEA as well, with only a few modifications listed below:

- Floor plan's quality assessment, line 2 and line 16: contrary to EA, in MOEA, multiple quality measures are calculated for each solution.
- Parental selection, line 7 and line 9: the EA uses a standard tournament selection that chooses the best competitor solution with respect to the single objective. In MOEA, the *crowded-comparison operator* is used within the tournament selection. It guides the selection process towards a uniformly-spread-out Pareto-optimal front by considering both the quality and uniqueness of the solutions; see [40].
- Forming the final $population$ in generation $g + 1$, line 18 and line 19: instead of the actions taken on line 18 and line 19 of the EA, MOEA adopts the procedure that prefers high-quality and unique solutions out of the set $tempPop \cup population$; see Figure 2 in [40].
- Selection of the final solution(s), line 21: in EA, a single best-of-run solution is returned. In MOEA, the whole set of all non-dominated solutions of the last population is returned.

References

1. Tompkins, J.; White, J.; Bozer, Y.; Tanchoco, J. *Facilities Planning*; Wiley: Hoboken, NJ, USA, 2010.
2. Papakostas, N.; O'Connor Moneley, J.; Hargaden, V. Integrated simulation-based facility layout and complex production line design under uncertainty. *CIRP Ann.* **2018**, *67*, 451–454. [CrossRef]
3. YounesSinaki, R.; Sadeghi, A.; Mosadegh, H.; Almasarwah, N.; Suer, G. Cellular manufacturing design 1996–2021: A review and introduction to applications of Industry 4.0. *Int. J. Prod. Res.* **2022**, 1–52.
4. Izadinia, N.; Eshghi, K. A robust mathematical model and ACO solution for multi-floor discrete layout problem with uncertain locations and demands. *Comput. Ind. Eng.* **2016**, *96*, 237–248. [CrossRef]
5. Zhang, Z.; Wu, L.; Wu, Z.; Zhang, W.; Jia, S.; Peng, T. Energy-Saving Oriented Manufacturing Workshop Facility Layout: A Solution Approach Using Multi-Objective Particle Swarm Optimization. *Sustainability* **2022**, *14*, 2788. [CrossRef]
6. Drira, A.; Pierreval, H.; Hajri-Gabouj, S. Facility layout problems: A survey. *Annu. Rev. Control.* **2007**, *31*, 255–267. [CrossRef]
7. Hosseini-Nasab, H.; Fereidouni, S.; Ghomi, S.M.T.F.; Fakhrzad, M.B. Classification of facility layout problems: A review study. *Int. J. Adv. Manuf. Technol.* **2017**, *94*, 957–977. [CrossRef]
8. Pérez-Gosende, P.; Mula, J.; Díaz-Madroñero, M. Facility layout planning. An extended literature review. *Int. J. Prod. Res.* **2021**, *59*, 3777–3816. [CrossRef]
9. Guo, W.; Jiang, P.; Yang, M. Unequal area facility layout problem-solving: A real case study on an air-conditioner production shop floor. *Int. J. Prod. Res.* **2023**, *61*, 1479–1496. [CrossRef]
10. Moatari-Kazerouni, A.; Chinniah, Y.; Agard, B. Integrating occupational health and safety in facility layout planning, part I: Methodology. *Int. J. Prod. Res.* **2015**, *53*, 3243–3259. [CrossRef]
11. Koopmans, T.C.; Beckmann, M.J. Assignment Problems and the Location of Economic Activities. *Econometrica* **1957**, *25*, 53. [CrossRef]
12. Loiola, E.M.; de Abreu, N.M.M.; Boaventura-Netto, P.O.; Hahn, P.; Querido, T. A survey for the quadratic assignment problem. *Eur. J. Oper. Res.* **2007**, *176*, 657–690. [CrossRef]
13. Ejeh, J.O.; Liu, S.; Papageorgiou, L.G. Optimal multi-floor process plant layout with production sections. *Chem. Eng. Res. Des.* **2018**, *137*, 488–501. [CrossRef]
14. Klausnitzer, A.; Lasch, R. Optimal facility layout and material handling network design. *Comput. Oper. Res.* **2019**, *103*, 237–251. [CrossRef]
15. Vázquez-Román, R.; Díaz-Ovalle, C.O.; Jung, S.; Castillo-Borja, F. A reformulated nonlinear model to solve the facility layout problem. *Chem. Eng. Commun.* **2019**, *206*, 476–487. [CrossRef]

16. Gülşen, M.; Murray, C.C.; Smith, A.E. Double-row facility layout with replicate machines and split flows. *Comput. Oper. Res.* **2019**, *108*, 20–32. [CrossRef]
17. Baykasoglu, A.; Subulan, K.; Hamzadayı, A. Capability-based machine layout with a matheuristic-based approach. *Expert Syst. Appl.* **2022**, *198*, 116900. [CrossRef]
18. Scholz, D.; Petrick, A.; Domschke, W. STaTS: A Slicing Tree and Tabu Search based heuristic for the unequal area facility layout problem. *Eur. J. Oper. Res.* **2009**, *197*, 166–178. [CrossRef]
19. Komarudin.; Wong, K.Y. Applying Ant System for solving Unequal Area Facility Layout Problems. *Eur. J. Oper. Res.* **2010**, *202*, 730–746. [CrossRef]
20. Wong, K.Y.; Komarudin. Solving facility layout problems using Flexible Bay Structure representation and Ant System algorithm. *Expert Syst. Appl.* **2010**, *37*, 5523–5527. [CrossRef]
21. Derakhshan Asl, A.; Wong, K.Y. Solving Unequal-area Static and Dynamic Facility Layout Problems Using Modified Particle Swarm Optimization. *J. Intell. Manuf.* **2017**, *28*, 1317–1336. [CrossRef]
22. Guan, J.; Lin, G. Hybridizing variable neighborhood search with ant colony optimization for solving the single row facility layout problem. *Eur. J. Oper. Res.* **2016**, *248*, 899–909. [CrossRef]
23. Saraswat, A.; Venkatadri, U.; Castillo, I. A framework for multi-objective facility layout design. *Comput. Ind. Eng.* **2015**, *90*, 167–176. [CrossRef]
24. Aiello, G.; Scalia, G.L.; Enea, M. A multi objective genetic algorithm for the facility layout problem based upon slicing structure encoding. *Expert Syst. Appl.* **2012**, *39*, 10352–10358. [CrossRef]
25. García-Hernández, L.; Arauzo-Azofra, A.; Salas-Morera, L.; Pierreval, H.; Corchado, E. Facility Layout Design Using a Multi-objective Interactive Genetic Algorithm to Support the DM. *Expert Syst. J. Knowl. Eng.* **2015**, *32*, 94–107. [CrossRef]
26. Gonçalves, J.F.; Resende, M.G. A biased random-key genetic algorithm for the unequal area facility layout problem. *Eur. J. Oper. Res.* **2015**, *246*, 86–107. [CrossRef]
27. Liu, J.; Liu, J. Applying multi-objective ant colony optimization algorithm for solving the unequal area facility layout problems. *Appl. Soft Comput.* **2019**, *74*, 167–189. [CrossRef]
28. Garcia-Hernandez, L.; Garcia-Hernandez, J.; Salas-Morera, L.; Carmona-Muñoz, C.; Alghamdi, N.; de Oliveira, J.V.; Salcedo-Sanz, S. Addressing Unequal Area Facility Layout Problems with the Coral Reef Optimization algorithm with Substrate Layers. *Eng. Appl. Artif. Intell.* **2020**, *93*, 103697. [CrossRef]
29. Krömer, P.; Platoš, J.; Snášel, V. Solving the Single Row Facility Layout Problem by Differential Evolution. In Proceedings of the 2020 Genetic and Evolutionary Computation Conference, GECCO '20, New York, NY, USA, 8–12 July 2020; pp. 210–218. [CrossRef]
30. García-Hernández, L.; Palomo-Romero, J.M.; Salas-Morera, L.; Arauzo-Azofra, A.; Pierreval, H. A novel hybrid evolutionary approach for capturing decision maker knowledge into the unequal area facility layout problem. *Expert Syst. Appl.* **2015**, *42*, 4697–4708. [CrossRef]
31. Reisinger, J.; Zahlbruckner, M.A.; Kovacic, I.; Kán, P.; Wang-Sukalia, X.; Kaufmann, H. Integrated multi-objective evolutionary optimization of production layout scenarios for parametric structural design of flexible industrial buildings. *J. Build. Eng.* **2022**, *46*, 103766. [CrossRef]
32. Kubalík, J.; Kadera, P.; Jirkovský, V.; Kurilla, L.; Prokop, Š. Plant Layout Optimization Using Evolutionary Algorithms. In *Industrial Applications of Holonic and Multi-Agent Systems: 9th International Conference, HoloMAS 2019, Linz, Austria, 26–29 August 2019*; Mařík, V., Kadera, P., Rzevski, G., Zoitl, A., Anderst-Kotsis, G., Tjoa, A.M., Khalil, I., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 173–188.
33. Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by Simulated Annealing. *Science* **1983**, *220*, 671–680. [CrossRef]
34. Anand, S.; Saravanasankar, S.; Subbaraj, P. Customized simulated annealing based decision algorithms for combinatorial optimization in VLSI floorplanning problem. *Comput. Optim. Appl.* **2012**, *52*, 667–689. [CrossRef]
35. Yadav, A.; Jayswal, S. Makespan minimization in flexible manufacturing system using genetic algorithm. *Int. J. Appl. Eng. Res.* **2018**, *13*, 9328–9334.
36. Umbarkar, A.; Sheth, P.D. Crossover Operators in Genetic Algorithms: A Review. *ICTACT J. Soft Comput.* **2015**, *6*, 1083–1092
37. Hill, S.; O'Riordan, C. Examining the impact of Neutral theory on Genetic Algorithm population evolution. In Proceedings of the 2015 7th International Joint Conference on Computational Intelligence (IJCCI), Lisbon, Portugal, 12–14 November 2015; Volume 1, pp. 196–203.
38. Galván-López, E.; Poli, R.; Kattan, A.; O'Neill, M.; Brabazon, A. Neutrality in evolutionary algorithms... What do we know? *Evol. Syst.* **2011**, *2*, 145–163. [CrossRef]
39. Knowles, J.; Corne, D. On metrics for comparing nondominated sets. In Proceedings of the 2002 Congress on Evolutionary Computation, CEC'02 (Cat. No.02TH8600), Honolulu, HI, USA, 12–17 May 2002; Volume 1, pp. 711–716. [CrossRef]
40. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]