Fair optimization with advanced aggregation operators in a multicriteria facility layout problem

Jarosław Hurkała
Warsaw University of Technology
Institute of Control & Computation Engineering
ul. Nowowiejska 15/19, 00-665 Warsaw, Poland
Email: J.Hurkala@elka.pw.edu.pl

Adam Hurkała
Warsaw University of Technology
Institute of Control & Computation Engineering
ul. Nowowiejska 15/19, 00-665 Warsaw, Poland
Email: A.Hurkala@elka.pw.edu.pl

***Abstract*—In this paper we address a mining operation problem that is a special case of Quadratic Assignment Problem and which belongs to the class of facility layout problems. The considered problem is static and discrete, but the set of possible locations is larger than the set of facilities. We distinguish multiple types of equal-area facilities (mines, processing and auxiliary facilities). Mines can be placed only on selected locations (deposits of various resources), and the production volume of each type of facility depends on the adjacency of other facilities. We examine two situations: when the number of each type of facility is given, and when only the total number of facilities is specified. The goal is to maximize the production. This problem is multi-objective and we use advanced aggregation operators (OWA/WOWA) to achieve fair solutions. A comparison of results obtained with list-based threshold accepting meta-heuristic and simulated annealing algorithm is presented.**

I. Introduction

THE facility layout is a broad class of problems that has been a subject of interest for researches around the world due to many real-life applications as well as significant scientific value. Layout problems are generally NP-Hard [1], which makes them perfect test ground for various optimization algorithms and heuristic approaches. Because of different assumptions and vast variety of considerations, it is difficult to arrive at one, common definition that would encompass all possible kinds of problems associated with finding an optimal placement of facilities in some predefined area.

In terms of layout evolution, the problem can be static, i.e. the key information about the problem (e.g. parameters and relationship between facilities) are constant, or dynamic, when possible changes over subsequent time periods additionally have to be taken into account (see [2] for both formulations).

By analyzing the layout formulations, we can classify the problems as continuous, in which the facilities can be placed anywhere within the designated area [3], or discrete, when the facilities can be placed only in specified locations [4]. The discrete formulation can be considered as a Quadratic Assignment Problem (QAP) [5], in which the assignment function is a bijection, i.e. the number of facilities is equal to the number of possible locations. Moreover, in QAP for each pair of locations a distance is specified and for each pair of facilities a weight is given.

In the literature we can find many different models with single criterion, e.g. total material handling cost, travel time of parts, travel distance, and problems with multiple criteria, for which at the same time different objectives are minimized, e.g. material handling, tools and information flow. For the latter problems, most researchers use a simple aggregation function, e.g. a weighted sum ([6], [7]).

In a great number of articles about the facility layout problems a meta-heuristic is chosen to solve the underlaying optimization problem. The evolutionary algorithms, and most notably genetic algorithms, are the most popular choice [3], [4], [8], [9], [10], [11]. The other group of approximated approaches that attract the interest of researchers are the random search methods - tabu search ([12], [13]) and simulated annealing ([14], [15]).

In this paper we address a layout problem that is static and discrete, but contrary to QAP the considered assignment function is not a bijection. Moreover, we define the relationship between the facilities and their location in a more complex way.

The goal of the facility layout problem considered in this paper is to find a non-overlapping planar arrangement of n rectangular facilities within a given rectangular site that maximizes the production of facilities, and unlike [11], [13] we assume equal-area facilities.

Furthermore, we distinguish multiple types of facilities (and therefore multiple types of products), hence the problem is multicriterial. We show that the application of the ordered weighted averaging (OWA) aggregation introduced by [16] is a consistent, reasonable and fairness-preserving approach to model a multicriteria facility layout problem.

To solve the problem, we propose a novel (not mentioned in the recent reviews and surveys on facility layout problems [17], [18]) heuristic approach based on the threshold accepting algorithm – a list-based threshold accepting meta-heuristic, that was successfully applied in a job-shop scheduling problem [19]. We compare the effectiveness of this algorithm with our design of simulated annealing ([20], [27]).

The paper is organized as follows. In Section 2, we present the definition of the multicriteria facility layout problem that we discuss in this article. Sections 3 briefly portrays the fair aggregation operators. In Section 4, we describe list-based threshold accepting and simulated annealing algorithms, and explain the implementation details for the problem at hand.

The results of the experiments are shown in Section 5, and the concluding remarks are presented in Section 6.

II. Problem definition

A. Basic notation and definitions

The facility layout problem that we discuss in this paper can be simply described as a problem of choosing a location $l \in \mathcal{L}$ for each facility $f \in \mathcal{F}$ so that the objective function (that will be defined later on) is maximized.

Remark 2.1: We assume equal-area facilities which are placed within a rectangular $n \times m$ grid. Each location is thus a cell in the grid and the considered problem is combinatorial in nature.

Remark 2.2: We assume $|\mathcal{L}| > |\mathcal{F}|$ so that the problem does not boil down to simply finding a correct permutation of facility locations. In consequence, the assignment function is not a bijection, as in QAP formulation.

We distinguish three main classes of facilities : $\mathcal{F} = \mathcal{P} \cup \mathcal{A} \cup \mathcal{M}$, where $\mathcal{P} = \bigcup_{t \in \mathcal{T}} \mathcal{P}_t$ is the set of $|\mathcal{T}|$ types of processing facilities, $\mathcal{A} = \bigcup_{v \in \mathcal{V}} \mathcal{A}_v$ is the set of $|\mathcal{V}|$ types of auxiliary facilities, and $\mathcal{M} = \bigcup_{u \in \mathcal{U}} \mathcal{M}_u$ denotes the set of $|\mathcal{U}|$ types of mines.

Remark 2.3: We assume $|\mathcal{T}| > 1$ and $|\mathcal{U}| > 1$, i.e. there are at least one processing facility and at least one mine.

Corollary 1: The problem is multicriterial in nature.

Remark 2.4: Throughout this paper we will commonly denote $\mathcal{C} = \mathcal{T} \cup \mathcal{U}$ as the set of all the facility types that are responsible for production, and $\mathcal{D} = \mathcal{T} \cup \mathcal{V} \cup \mathcal{U}$ as the set of all possible facility types.

The set of locations is divided into two subsets $\mathcal{L} = \mathcal{S} \cup \mathcal{E}$, respectively locations $\mathcal{S}$ for processing/auxiliary facilities, and extraction sites $\mathcal{E} = \bigcup_{r \in \mathcal{R}} \mathcal{E}_r$ where deposits of $|\mathcal{U}|$ types of resources $\mathcal{R} = \bigcup_{u \in \mathcal{U}} \mathcal{R}_u$ are located, and where the corresponding types of mines can be placed.

Remark 2.5: We assume that $\mathcal{S} \cap \mathcal{E} = \emptyset$ (i.e. a processing/auxiliary facility cannot be placed on an extraction site and a mine can be located only at one of the resource deposits).

In our facility layout problem the objective is to maximize the production output of processing facilities and mines.

Definition 1 (Basic production output): The basic production output denoted by $O_c^B, c \in \mathcal{C}$ is a production volume of a specified type of facility (processing facility or mine) regardless of its location and the locations of other facilities.

The auxiliary facilities by definition are not producing anything themselves, nevertheless they play an important role by affecting production of other facilities. Moreover, we assume that resource deposits have an impact on the facilities production as well and that in general the facilities themselves can positively affect each other.

Definition 2 (Distance function): The relationship between facilities is based on a binary distance function $d : \mathcal{L} \times \mathcal{L} \to \{0,1\}$, i.e. either a pair of locations is adjacent or not. The facilities are placed in cells of a rectangular grid, and thus we assume that a relationship exists only between two adjacent facilities and facilities adjacent to resource deposits.

Definition 3 (Adjacent location): The set of all locations adjacent to a location $l \in \mathcal{L}$ will be denoted by $\delta_l(l) \subset \mathcal{L}$.

Definition 4 (Adjacent facility): The set of all facilities of type $d \in \mathcal{D}$ adjacent to a facility placed on location l will be denoted by $\delta_d^F(l) \subset \mathcal{F}$.

Definition 5 (Adjacent resource deposit): The set of all resource deposits of type $u \in \mathcal{U}$ adjacent to a facility f placed on location l will be denoted by $\delta_u^R(l) \subset \mathcal{R}$.

Definition 6 (Extra production output - facility): The extra production output denoted by $O_{cd}^F : \mathbb{N} \to \mathbb{R}$ is an additional production volume of a processing facility/mine $c \in \mathcal{C}$ proportional to the number of surrounding (adjacent) facilities $d \in \mathcal{D}$.

Definition 7 (Extra production output - resource deposit): The extra production output denoted by $O_{cu}^R : \mathbb{N} \to \mathbb{R}$ is an additional production volume of a processing facility/mine $c \in \mathcal{C}$ proportional to the number of surrounding (adjacent) resource deposits $u \in \mathcal{U}$.

Definition 8 (Processing facility production output): The (final) production output of a processing facility is defined as follows:

$$o_c(l) = O_c^B \left(\sum_{d \in \mathcal{D}} O_{cd}^F(\delta_d^F(l)) + \sum_{u \in \mathcal{U}} O_{cu}^R(\delta_u^R(l)) \right) \quad c \in \mathcal{C} \tag{1}$$

B. Optimization model

The mining operation problem considered in this article can be formulated as follows:

$$\max_x \; [o_1, o_2, \ldots, o_{|\mathcal{C}|}] \tag{2}$$

$$o_c^B = \sum_{l \in \mathcal{L}} O_c^B x_{lc} \quad c \in \mathcal{C} \tag{3}$$

$$o_c^F = \sum_{l \in \mathcal{L}} \sum_{i \in \delta(l)} \sum_{j \in \mathcal{D}} O_{cj}^F x_{ij} \quad c \in \mathcal{C} \tag{4}$$

$$o_c^R = \sum_{l \in \mathcal{L}} \sum_{i \in \delta(l)} \sum_{j \in \mathcal{U}} O_{cj}^R x_{ij} \quad c \in \mathcal{C} \tag{5}$$

$$o_c = o_c^B (o_c^F + o_c^R) \quad c \in \mathcal{C} \tag{6}$$

$$\sum_{c \in \mathcal{C}} x_{cl} = 1 \quad l \in \mathcal{L} \tag{7}$$

$$\sum_{l \in \mathcal{L}} \sum_{c \in \mathcal{C}} x_{cl} = |\mathcal{F}| \tag{8}$$

$$\sum_{l \in \mathcal{L}} x_{cl} = \alpha_c \quad c \in \mathcal{C} \tag{9}$$

$$x_{cl} \in \{0,1\} \quad c \in \mathcal{C}, l \in \mathcal{L} \tag{10}$$

where α_c is a parameter that denotes the designated number of each facility type (the constraint (9) is optional).

III. Ordered weighted averaging

In the ordered weighted averaging aggregation of outcomes (OWA) $\mathbf{y} = (y_1, \ldots, y_m)$ the weights $\mathbf{w} = (w_1, w_2, \ldots, w_m)$ are assigned to the ordered values (i.e., to the smallest value, the second smallest and so on) rather than to the specific criteria:

$$A_w = \sum_{i=1}^{m} w_i \theta_i(\mathbf{y}) \tag{11}$$

where $(\theta_1(\mathbf{y}), \theta_2(\mathbf{y}), \ldots, \theta_m(\mathbf{y}) = \Theta(\mathbf{y})$ is the ordering map $R^m \to R^m$ with $\theta_1(\mathbf{y}) \leq \theta_2(\mathbf{y}) \leq \ldots \leq \theta_m(\mathbf{y})$ and there exists a permutation τ of set I such that $\theta_i(\mathbf{y}) = y_{\tau(i)}$ for $i = 1, 2, \ldots, m$. The OWA operator provides a parameterized family of aggregation operators, which include many of the well-known operators such as the maximum, the minimum, the k-order statistics (including Conditional Value at Risk), the median and the arithmetic mean. The OWA satisfies the properties of strict monotonicity, impartiality and, in the case of monotonic increasing weights $w_1 > w_2 > \ldots > w_{m-1} > w_m$, the property of equitability [21] (satisfies the principle of transfers – equitable transfer of an arbitrary small amount from the larger outcome to a smaller outcome results in a more preferred achievement vector). Every solution maximizing the OWA function is then an equitably efficient solution to the original multiple criteria problem. Moreover, for linear multiple criteria problems every equitably efficient solution can be found as an optimal solution to the OWA aggregation with appropriate weights. Thus the OWA-based optimization generates the so-called equitably efficient solutions (cf. [22] for the formal axiomatic definition). According to [22] and [23], equitable efficiency expresses the concept of fairness, in which all system entities have to be treated equally and in the stochastic problems equitability corresponds to the risk aversion [24].

For the facility layout problem (2)–(10) the final OWA aggregation of the outcomes p_i for all types of production facilities $i \in \mathcal{C}$ can be stated as the following LP model:

$$\max \sum_{k=1}^{|\mathcal{C}|} k w'_k t_k - \sum_{k=1}^{|\mathcal{C}|} \sum_{c \in \mathcal{C}} w'_k d_{ck} \tag{12}$$

subject to

$$d_{ck} \geq t_k - o_c,\ d_{ck} \geq 0 \quad k = 1, 2, \ldots, |\mathcal{C}|,\ c \in \mathcal{C} \tag{13}$$

$$\mathbf{o} \in O \tag{14}$$

where coefficients w'_i are defined as $w'_m = w_m$ and $w'_i = w_i - w_{i+1}$ for $i = 1, 2, \ldots, m-1$, $\mathbf{o} = [o_c]_{c \in \mathcal{C}}$ and O is a feasible set of production output vectors defined by (3)–(10).

The weighted ordered weighted averaging (WOWA) aggregation is a generalization of the OWA aggregation, that allows assigning importance weights to specific criteria [25]. Those weights could express, for example, relative importance of different facility types. The weights assigned to ordered values will be further called preferential weights.

Let $\mathbf{p} = (p_1, \ldots, p_m)$ be an m-dimensional vector of importance weights such that $p_i \geq 0$ for $i = 1, \ldots, m$ and $\sum_{i=1}^{m} p_i = 1$. The corresponding Weighted OWA aggregation of vector $\mathbf{y}$ is defined [11] as follows:

$$A_{w,p} = \sum_{i=1}^{m} \omega_i \theta_i(\mathbf{y}) \tag{15}$$

with

$$\omega_i = w^*(\sum_{k \leq i} p_{\tau(k)}) - w^*(\sum_{k < i} p_{\tau(k)}), \tag{16}$$

where w^* is an increasing function interpolating points $(i/m, \sum_{k \leq i} w_k)$ together with the point $(0, 0)$ and τ representing the ordering permutation for $\mathbf{y}$ (i.e. $y_{\tau(i)} = \theta(\mathbf{y})$). Moreover, function w^* is required to be a straight line when the points can be interpolated in this way. We assume the piecewise linear interpolation function w^* which is the simplest form of the required interpolation.

Note, that the piecewise linear functions may be built with various number of breakpoints, not necessarily equal to number of criteria m [25]. Thus, any nonlinear function can be well approximated by a piecewise linear function with appropriate number of breakpoints. Therefore, we will consider weights vectors $\mathbf{w}$ of dimension n not necessarily equal to m. It is even possible to define a generalized WOWA aggregation where the preferential weights w_k are allocated to an arbitrarily defined grid of ordered outcomes defined by quantile breakpoints (see [25] and references therein).

As shown in [25], maximization of an equitable WOWA aggregation with decreasing preferential weights $w_1 \geq w_2 \geq \ldots \geq w_n$ may be implemented as the LP expansion of the original problem. In the case of the facility layout problem (2)–(10), this can be stated as follows:

$$\max \sum_{k=1}^{n} w'_k \left[\frac{k}{n} t_k - \sum_{c \in \mathcal{C}} p_c d_{ck} \right] \tag{17}$$

subject to

$$d_{ck} \geq t_k - o_c,\ d_{ck} \geq 0 \quad k = 1, 2, \ldots, n,\ c \in \mathcal{C} \tag{18}$$

$$\mathbf{o} \in O \tag{19}$$

If the importance weights are equal $p_c = 1/|\mathcal{C}|$, the model reduces to the OWA aggregation.

IV. Algorithms

A. List-based threshold accepting

List-based threshold accepting algorithm (LBTA) [19] is an extent of threshold accepting meta-heuristic, which belongs to the randomized search class of algorithms. The search trajectory crosses the solution space by moving from one solution to a random neighbor of that solution, and so on. Unlike the greedy local search methods which consist of choosing a better solution from the neighborhood of the current solution until such can be found (hill climbing), the threshold accepting allows choosing a worse candidate solution based on a threshold value. In the general concept of the threshold accepting algorithm it is assumed that a set of decreasing threshold values is given before the computation or an initial threshold value and a decrease schedule is specified.

Algorithm 1 Creating the list of threshold values

```
Require: Initial solution $s_1$, list size $S$, set of move operators $m \in M$
 1: $i \leftarrow 0$
 2: while $i < N$ do
 3:     $m \leftarrow random(M)$
 4:     $s_2 \leftarrow m(s_1)$
 5:     if $C(s_1) \leq C(s_2)$ then
 6:         $\Delta \leftarrow (C(s_2) - C(s_1))/C(s_1)$
 7:         $list \leftarrow list \cup \{\Delta\}$
 8:         $i \leftarrow i + 1$
 9:     else
10:         $s_1 \leftarrow s_2$
11:     end if
12: end while
13: return $list$
```

The rate at which the values decrease controls the trade-off between diversification (associated with large threshold values) and intensification (small threshold values) of the search. It is immensely difficult to predict how the algorithm will behave when a certain decrease rate is applied for a given problem without running the actual computation. It is also very common that the algorithm with the same parameters works better for some problem instances and significantly worse for others. These reflections led to the list-based threshold accepting branch of threshold accepting meta-heuristic.

In the list-based threshold accepting approach, instead of a predefined set of values, a list is dynamically created during a presolve phase of the algorithm. The list, which in a way contains knowledge about the search space of the underlying problem, is then used to solve it.

1) Creating the list of threshold values: The first phase of the algorithm consists of gathering information about the search space of the problem that is to be solved. From an initial solution a neighbor solution is created using a move function (perturbation operator) chosen at random from a predefined set of functions. If the candidate solution is better than the current one, it is accepted and becomes the current solution. Otherwise, a threshold value is calculated as a relative change between the two solutions:

$$\Delta = (C(s_2) - C(s_1))/C(s_1) \tag{20}$$

and added to the list, where $C(s_i)$ is the objective function value of the solution $s_i \in S$, and S is a set of all feasible solutions. For this formula to work, it is silently assumed that $C : S \to \mathbb{R}_+ \cup \{0\}$. This procedure is repeated until the specified size of the list is reached. For the algorithm overview see Algorithm 1.

2) Optimization procedure: The second phase of the algorithm is the main optimization routine, in which a solution to the problem is found. The algorithm itself is very similar to that of the previous phase. We start from an initial solution, create new solution from the neighborhood of current one using one of the move function, and compare both solutions.

Algorithm 2 LBTA optimization procedure

```
Require: Initial solution $s_1$, thresholds list $L$, set of move operators $m \in M$
 1: $i \leftarrow 0$
 2: $s^* \leftarrow s_1$
 3: while $i \leq N$ do
 4:     $m \leftarrow random(M)$
 5:     $s_2 \leftarrow m(s_1)$
 6:     $i \leftarrow i + 1$
 7:     if $C(s_2) \leq C(s_1)$ then
 8:         if $C(s_2) \leq C(s^*)$ then
 9:             $s^* \leftarrow s_2$
10:         end if
11:         $s_1 \leftarrow s_2$
12:         $i = 0$
13:     else
14:         $\Delta_{new} \leftarrow (C(s_2) - C(s_1))/C(s_1)$
15:         if $\Delta_{new} < max(list)$ then
16:             $list \leftarrow list \setminus \{max(list)\}$
17:             $list \leftarrow list \cup \{\Delta_{new}\}$
18:             $s_1 \leftarrow s_2$
19:             $i = 0$
20:         end if
21:     end if
22: end while
23: return $list$
```

If the candidate solution is better, it becomes the current one. Otherwise a relative change is calculated. To this point algorithms in both phases are identical. The difference in the optimization procedure is that we compare the threshold value with the largest value from the list. If the new threshold value is larger, then the new solution is discarded. Otherwise, the new threshold value replaces the value from the list, and the candidate solution is accepted to next iteration. The best solution found during the optimization process is considered final.

The list-based threshold accepting algorithm also incorporates early termination mechanism: after a (specified) number of candidate solutions is subsequently discarded, the optimization is stopped, and the best solution found so far is returned.

The optimization procedure of the list-based threshold accepting algorithm is shown in Algorithm 2.

B. Simulated annealing

The optimization process of the simulated annealing algorithm can be described in the following steps. At the start, an initial solution is required. Then, repeatedly, a candidate solution is randomly chosen from the neighborhood of the current solution. If the candidate solution is the same or better than the current one, it is accepted and replaces the current solution. Even if the generated solution is worse than the current one, it still has a chance to be accepted with, so called, acceptance probability. This probability is a function of difference between objective value of the current and

Algorithm 3 Simulated Annealing

```
Require: Initial solution $s_1$
1: $s^* \leftarrow s_1$
2: for $i = 1$ to $N$ do
3:   for $t = 1$ to $N_{const}$ do
4:     $s_2 \leftarrow perturbate(s_1)$
5:     $\delta \leftarrow C(s_2) - C(s_1)$
6:     if $\delta \leq 0$ or $e^{-\delta/k\tau} > random(0,1)$ then
7:       $s_1 \leftarrow s_2$
8:     end if
9:     if $C(s_2) < C(s^*)$ then
10:      $s^* \leftarrow s_2$
11:    end if
12:  end for
13:  $\tau \leftarrow \tau * \alpha$
14: end for
15: return $s^*$
```

the candidate solution and depends on a control parameter taken from the thermodynamics, called temperature. The temperature is decreased after a number of iterations, and the process continues as described above. The optimization is stopped either after a maximum number of iterations or when a minimum temperature is reached. The best solution found during the annealing process is considered final.

For the algorithm overview see Algorithm 3.

In order to apply the simulated annealing algorithm to the facility layout problems, the annealing process must be adapted and the parameters adjusted appropriately. Similarly to the threshold decrease rate in LBTA, the temperature decrease (also known as the cooling process) in simulated annealing consists of decreasing the temperature by a so called reduce factor. The parameters associated with this mechanism are:

1) Initial temperature.
2) Function of temperature decrease in consecutive iterations.
3) The number of iterations at each temperature (Metropolis equilibrium).
4) Minimum temperature at which the algorithm terminates or alternatively the maximum number of iterations as the stopping criterion.

Let τ be the temperature and α be the reduce factor. Then the annealing scheme can be represented as the following recursive function:

$$\tau^{i+1} = \alpha * \tau^i, \tag{21}$$

where i is the number of current iteration in which the cooling schedule takes place.

Second building block of SA that has to be customized is the acceptance probability function, which determines whether to accept or reject candidate solution that is worse than the current one. The most widely used function is:

$$p(\delta, \tau) = e^{-\delta/k\tau}, \tag{22}$$

TABLE I
SIMULATED ANNEALING PARAMETERS

Parameter	Description	Value
α	Reduce factor	$1 - \frac{5}{N}$
τ^0	Initial temperature	0.99
δ^0	Minimal difference between solutions	1
p^0	Initial acceptance probability	1
N_{const}	Number at each constant temperature	10
N	Number of SA iterations	100000

where $\delta = E(s_2) - E(s_1)$ is the difference between the objective value (denoted by E) of the candidate (s_2) and the current solution (s_1), and k is the Boltzmann constant found by:

$$k = \frac{\delta^0}{\log \frac{p^0}{\tau^0}}, \tag{23}$$

where δ^0 is an estimated difference between objective values of two solutions, p^0 is the initial value of the acceptance probability and τ^0 is the initial temperature. Notice that we use decimal logarithm rather than natural, which is most widely seen in the literature and, rather than average, we use minimal difference between solutions.

For the overview of the parameters applied in the facility layout problem, see Table I.

C. Neighborhood function

The most problem-specific mechanism of both the SA and the LBTA algorithm that always needs a different approach and implementation is the procedure of generating a candidate solution from the neighborhood of the current one, which is called a perturbation scheme, transition operation/operator or a move function. Although there are many ways to accomplish this task, we have examined the following techniques:

1) Interchange two adjacent processing facilities.
2) Interchange two processing facilities at random locations.
3) Move a single processing facility to an adjacent, empty location.
4) Change type of the facility.
5) Move a mine from current resource deposit to another deposit that is not occupied.

In order to generate a new solution, the LBTA algorithm applies one of the aforementioned operators chosen at random to the current solution. SA on the other hand uses only one, compound operator (a combination of operators which together allow to make a transition from initial to any feasible solution) during the whole optimization procedure.

D. Implementation details

1) Zero elements: In the first phase of the list-based threshold accepting algorithm the list is populated with values of relative change between two solutions $\Delta \geq 0$. After careful consideration, we believe that including zeros in the list is a misconception. In the actual optimization procedure, i.e. the second phase, the threshold value is computed only if the new solution is worse than the current one, which means that the calculated relative change will always have a positive value ($\Delta_{new} > 0$). The new threshold value is compared with the largest value from the list (T_{hmax}). Thus, we can distinguish three cases:

1) $T_{hmax} = 0$: since thresholds are non-negative from definition, in this case the list contains all zero elements and it will not change throughout the whole procedure (T_{hmax} is constant). Comparing a positive threshold value Δ_{new} against zero yields in discarding the candidate solution. The conclusions are as follows:
   a) it does not matter how many zeros are in the list, the effective size of the list is equal to one,
   b) the algorithm is reduced to hill climbing algorithm that accepts candidate solutions which are at least as good as the current one.
2) $T_{hmax} > 0$ and $\Delta_{new} < T_{hmax}$: the largest (positive) threshold value from the list T_{hmax} is replaced by a smaller (positive) threshold value Δ_{new}. The number of zero elements in the list remains the same throughout the whole procedure and therefore is completely irrelevant to the optimization process. The effective list size is equal to the number of positive elements.
3) $T_{hmax} > 0$ and $\Delta_{new} \geq T_{hmax}$: the new solution is discarded and the list remains unchanged.

The main idea behind the list is to control the diversification and intensification of the search process. In the early stage of the search the algorithm should allow to cover as much solution space as possible, which means that the thresholds in the list are expected to be large enough to make that happen. In the middle stage, the algorithm should slowly stop fostering the diversification and begin to foster the intensification of the search. In the end stage, the intensification should be the strongest, i.e. the list is supposed to contain smaller and smaller threshold values, which induces discarding of worse solution candidates. In consequence, the algorithm is converging to a local or possibly even a global optimum.

2) Stopping criterion: Even though equipped with an early-termination mechanism, the LBTA algorithm does not have a solution space independent stopping criterion. If the number of subsequently discarded worse solutions is set too high, the algorithm will run for an unacceptable long time (it has been observed during preliminary tests). Hence, we propose to use a global counter of iterations so that when a limit is reached, the algorithm will terminate gracefully.

V. Numerical experiments

The numerical experiments were performed on a number of randomly generated different size problems. Each instance is defined by the number of locations $|\mathcal{L}|$, which in turn determines the number of facilities: $|\mathcal{F}| = |\mathcal{L}|/2$, and the number of resource deposits: $|\mathcal{R}| = |\mathcal{L}|/5$ (the number of resource deposits of each type are equal or differ by one). However, the following settings were the same for every instance, regardless of the problem size:

1) the number of types of processing facilities $|\mathcal{T}| = 2$,
2) the number of types of auxiliary facilities $|\mathcal{V}| = 2$,
3) the number of types of mines/resources $|\mathcal{U}| = 2$,
4) the extra production output depending on adjacent facilities is defined for pairs of processing facility type t_i and auxiliary facility type v_i: $O^F_{t_i v_i} = 0.4, i = 1, 2$ and for one pair of processing facility types: $O^F_{t_1 t_2} = 0.2$ (the relationship is not symmetrical),
5) the extra production output depending on adjacent resource deposits is defined for pairs of processing facility type t_i and resource deposit type u_i: $O^R_{t_i u_i} = 0.5, i = 1, 2$,
6) the total basic production output is constant, and is randomly distributed among the (basic) production output of processing facilities and mines.

Algorithm 4 Random distribution algorithm

```
Require: Number of elements $N$, value to distribute $v$
 1: $list \leftarrow v$
 2: for $i = 1$ to $N$ do
 3:     $x_0 \leftarrow max(list)$
 4:     $list \leftarrow list \setminus \{x_0\}$
 5:     $x_1 \leftarrow random(x_0)$
 6:     $x_2 \leftarrow x_0 - x_1$
 7:     $list \leftarrow list \cup \{x_1\} \cup \{x_2\}$
 8: end for
 9: $list \leftarrow shuffle(list)$
10: return $list$
```

The algorithm of random distribution of total number of facilities among the facilities of each type, and distribution of total basic production output among the basic production output of different types of processing facilities and mines has been based on bisecting technique described in [26]. For the overview see Algorithm 4.

For the WOWA aggregation operator we have chosen already examined weights generation methodology ([27]): all the weights, except two, are strictly decreasing numbers with the step 0.1, while the two selected weights ($k = \lfloor n/3 \rfloor$ and $k = \lfloor 2n/3 \rfloor$) differ from the previous ones by 0.5.

After preliminary tests, based on the objective value and iteration number at which the best solution was found, we have arrived at the list size of 2000, for which the LBTA algorithm works at peak performance.

The algorithms were implemented in Java, and all the experiments were performed on a 3.1 GHz processor. The results are the average of 30 tries for each instance.

We have examined two situations: when the number of each type of facility is given (constraint (9)) – see Table II, and when only the total number of facilities is specified – the results are presented in Table III.

Both algorithms produced similar results in terms of both the solution quality (mean value and standard deviation) and computation time. For instances with fixed number of each facility type, SA has smaller standard deviation, but when it comes to instances with fixed total number of facilities, it is the other way around.

TABLE II
LAYOUT PROBLEMS WITH THE NUMBER OF EACH TYPE OF FACILITY GIVEN

$\lvert\mathcal{L}\rvert$	$\lvert\mathcal{F}\rvert$	$\lvert\mathcal{R}\rvert$	LBTA			SA			$LBTA/SA$
			mean objective value	standard deviation	time[s]	mean objective value	standard deviation	time[s]	
100	50	20	18719	19.1	1.745	18713	26.9	1.810	1.0004
100	50	20	11150	84.8	1.288	11205	76.8	1.824	0.9952
100	50	20	15406	6.4	1.688	15404	10.1	1.872	1.0001
100	50	20	11426	223.3	1.977	11612	142.3	2.039	0.9840
100	50	20	17599	48.6	1.700	17605	15.7	1.925	0.9997
144	72	29	27889	74.8	2.039	27882	80.7	2.261	1.0002
144	72	29	23739	139.4	2.008	23747	122.1	2.220	0.9997
144	72	29	27050	73.8	2.249	27055	72.0	2.438	0.9998
144	72	29	28103	49.8	2.114	28116	47.4	2.139	0.9995
144	72	29	19224	214.9	1.881	19626	205.5	2.405	0.9795
196	98	39	32146	237.0	2.572	32325	101.0	2.601	0.9945
196	98	39	23012	97.6	2.908	22993	70.0	3.281	1.0008
196	98	39	36957	81.6	2.346	36913	112.2	2.393	1.0012
196	98	39	34567	51.0	2.285	34563	51.2	2.690	1.0001
196	98	39	39903	90.7	2.943	39921	80.3	3.005	0.9995
256	128	51	57569	93.9	3.483	57629	46.8	4.155	0.9990
256	128	51	50916	167.1	3.763	51006	117.8	3.839	0.9982
256	128	51	59488	264.7	3.267	59658	189.7	3.565	0.9971
256	128	51	40648	141.4	2.061	40740	80.1	2.855	0.9977
256	128	51	33602	252.1	2.851	33744	189.2	3.022	0.9958
324	162	65	71843	122.3	4.149	71756	217.2	4.187	1.0012
324	162	65	26893	316.5	1.169	27174	148.2	2.455	0.9896
324	162	65	49630	126.3	3.617	49672	105.7	3.618	0.9992
324	162	65	43264	240.4	2.658	43347	144.9	2.908	0.9981
324	162	65	57886	401.5	3.583	58280	256.7	3.616	0.9933
400	200	80	92696	438.0	4.571	93413	267.3	5.027	0.9923
400	200	80	48824	288.3	3.068	49195	141.7	3.619	0.9925
400	200	80	66068	148.0	3.828	66403	172.8	3.874	0.9950
400	200	80	51355	382.8	3.612	51800	259.4	4.240	0.9914
400	200	80	72824	185.5	4.824	72848	168.5	5.159	0.9997

VI. CONCLUSION

We are not convinced that one algorithm supersedes the other, which only means that both heuristics are equally good and can be successfully applied to difficult, combinatorial problems, like the one considered in this paper.

By considering only the idea behind the algorithm, we find the LBTA meta-heuristic a little bit more appealing than SA because of the concept of the list. The list holds the key to control the optimization process. The larger the list, the longer the diversification stage will last, because there will be more threshold values to replace and higher probability of that happening. On the other hand, the smaller the list, the stronger the intensification, which in some cases can be more desired (e.g. when the algorithm cannot find a good solution within the given number of iterations). This gives the unique possibility to the algorithm designer to control the whole optimization process with just one parameter.

REFERENCES

[1] Garey, M. R.,& Johnson, D. S. "Computers and intractability: A guide to the theory of NP-completeness". *A Series of Books in the Mathematical Sciences.* W. H. Freeman and Company, 1979.

[2] Kouvelis, P., Kurawarwala, A. A., & Gutierrez, G. J. "Algorithms for robust single and multiple period layout planning for manufacturing systems". *European Journal of Operations Research*, 63(2), pp. 287–303, 1992.

[3] Dunker, T., Radonsb, G., & Westkämpera, E. "Combining evolutionary computation and dynamic programming for solving a dynamic facility layout problem". *European Journal of Operational Research*, 165(1), pp. 55–69, 2005.

[4] Fruggiero, F., Lambiase, A., & Negri, F. "Design and optimization of a facility layout problem in virtual environment". *In Proceeding of ICAD 2006*, pp. 2206–, 2006.

[5] Kaku, Bharat K.; Thompson, Gerald Luther; and Carnegie Mellon University Design Research Center., "An exact algorithm for the general quadratic assignment problem". *Tepper School of Business.* Paper 944. 1983.

[6] Chen, C. W., & Sha, D. Y. "Heuristic approach for solving the multi-objective facility layout problem". *International Journal of Production Research*, 43(21), pp. 4493–4507, 2005.

[7] Ye M., Zhou G. "A local genetic approach to multiobjective, facility layout problems with fixed aisles". *International Journal of Production Research*, 45, pp. 5243–5264, 2007.

[8] Pierreval, H., Caux, C., Paris, J. L., & Viguier, F. "Evolutionary approaches to the design and organization of manufacturing systems". *Computers & Industrial Engineering*, 44(3), pp. 339–364, 2003.

[9] Rajasekharan, M., Peters, B.A. and Yang, T. "A genetic algorithm for facility layout design in flexible manufacturing systems", *Int. J. Production Research*, Vol. 36, No. 1, pp. 95–110, 1998.

[10] Wang, M. J., Hu, M. H., & Ku, M. H. "A solution to the unequal area facilities layout problem by genetic algorithm". *Computers in Industry*, 56(2), pp. 207–220, 2005.

[11] Lee, Y. H., & Lee, M. H. "A shape-based block layout approach to facility layout problems using hybrid genetic algorithm". *Computers & Industrial Engineering*, 42, pp. 237–248, 2002.

[12] Abdinnour-Helm, S. and Hadley, S.W. "Tabu search based heuristics for multi-floor facility layout". *Int. J. Production Research*, Vol. 38, No. 2, pp. 365–383, 2000.

[13] McKendall, A.R. and Hakobyan, A. "Heuristics for the dynamic facility layout problem with unequal-area departments". *European Journal of Operational Research*, Vol. 201, No. 1, pp. 171–182, 2010.

[14] McKendall Jr., A.R., Shanga, J. and Kuppusamy, S. "Simulated annealing heuristics for the dynamic facility layout problem". *Computers and Operations Research*, Vol. 33, pp. 2431–2444, 2006.

TABLE III
LAYOUT PROBLEMS WITH THE TOTAL NUMBER OF FACILITIES GIVEN

$\|\mathcal{L}\|$	$\|\mathcal{F}\|$	$\|\mathcal{R}\|$	LBTA			SA			$LBTA/SA$
			mean objective value	standard deviation	time[s]	mean objective value	standard deviation	time[s]	
100	50	20	26406	41.5	2.298	26409	44.0	2.411	0.9999
100	50	20	24227	34.5	2.346	24235	27.3	2.400	0.9997
100	50	20	25111	32.6	2.344	25126	25.2	2.373	0.9994
100	50	20	26617	15.6	2.325	26599	40.2	2.397	1.0007
100	50	20	23024	41.2	2.286	23030	37.7	2.382	0.9998
144	72	29	38084	42.7	3.007	38120	41.0	3.032	0.9990
144	72	29	33482	23.6	2.969	33462	24.7	3.055	1.0006
144	72	29	34542	76.6	2.867	34490	92.8	3.178	1.0015
144	72	29	38675	56.3	2.877	38742	64.3	2.931	0.9982
144	72	29	39691	65.5	2.831	39754	63.1	2.928	0.9984
196	98	39	53543	112.8	3.153	53511	139.5	3.316	1.0006
196	98	39	51694	40.5	3.686	51748	71.0	3.742	0.9990
196	98	39	59454	51.9	3.689	59509	39.9	3.803	0.9991
196	98	39	52764	206.0	3.670	52714	207.3	3.912	1.0009
196	98	39	54205	64.0	3.538	54234	93.3	3.604	0.9995
256	128	51	68487	202.1	4.755	68456	227.9	4.827	1.0004
256	128	51	67708	49.7	4.532	67779	71.7	4.611	0.9990
256	128	51	92840	173.7	4.752	93035	204.4	4.838	0.9979
256	128	51	70598	71.8	4.533	70678	108.1	4.586	0.9989
256	128	51	56229	157.2	4.802	56245	143.4	4.821	0.9997
324	162	65	86166	110.4	5.564	86315	162.3	5.531	0.9983
324	162	65	77472	102.0	5.551	77480	118.2	5.476	0.9999
324	162	65	69985	51.4	5.562	70020	68.7	5.572	0.9995
324	162	65	90832	148.8	5.337	90801	192.5	5.378	1.0003
324	162	65	88273	185.5	5.412	88452	195.7	5.418	0.9980
400	200	80	146217	195.3	6.982	146485	299.3	6.992	0.9982
400	200	80	132636	273.7	7.096	132663	326.2	7.105	0.9998
400	200	80	124096	227.0	7.042	124082	402.6	7.063	1.0001
400	200	80	98076	155.2	6.243	98079	147.8	6.246	1.0000
400	200	80	86521	165.1	6.532	86574	94.7	6.648	0.9994

[15] Ioannou, G. “An integrated model and a decomposition-based approach for concurrent layout and material handling system design”. *Computers and Industrial Engineering*, Vol. 52, pp.459–485, 2007.

[16] Yager, R.R. “On ordered weighted averaging aggregation operators in multicriteria decision making”. IEEE Trans. Systems, Man and Cybernetics 18, pp. 183–190, 1988.

[17] Drira, A., Pierreval, H. and Hajri-Gabouj, S., “Facility layout problems: A survey”. *Annual Reviews in Control*, 31, pp. 255–267, 2007.

[18] Kundu, A. and Dan, P.K., “Metaheuristic in facility layout problems: current trend and future direction”. *Int. J. Industrial and Systems Engineering*, Vol. 10, No. 2, 2012.

[19] Lee, D.S., Vassiliadis, V.S., Park, J.M. “A novel threshold accepting meta-heuristic for the job-shop scheduling problem”. *Computers & Operations Research*, 31, pp. 2199–2213, 2004.

[20] Hurkała, J. and Hurkała, A. “Effective Design of the Simulated Annealing Algorithm for the Flowshop Problem with Minimum Makespan Criterion”. *9th International Conference on Decision Support for Telecommunications and Information Society*, September 5-6, 2011, Warsaw, Poland [Journal of Telecommunications and Information Technology, no. 2, 2012].

[21] Ogryczak, W., Śliwiński, T. “On Solving Linear Programs with the Ordered Weighted Averaging Objective”. *European Journal of Operational Research*, 148, pp. 80–91, 2003.

[22] Kostreva, M.M., Ogryczak, W., “Linear optimization with multiple equitable criteria”. *RAIRO Operations Research*, 33, pp. 275–297, 1999.

[23] Rawls, J., “The Theory of Justice”. *Cambridge: Harvard Univ Press*, 1971.

[24] Bell, D.E, Raiffa, H,. “Risky choice revisited, in Bell at all, Decision Making Descriptive, Normative and Prescriptive Interactions”. *Cambridge University Press*, Cambridge, pp. 99–112, 1988.

[25] Ogryczak, W., Śliwiński, T., “On Efficient WOWA Optimization for Decision Support under Risk”. *International Journal of Approximate Reasoning*, 50, pp. 915–928, 2009.

[26] M. Steinbach, G. Karypis, V. Kumar, “A Comparison of Document Clustering Techniques”. *KDD workshop on text mining*, 34, pp. 109–111, 2000.

[27] Hurkała, J. and Śliwiński, T., “Fair flow optimization with advanced aggregation operators in Wireless Mesh Networks”, *Federated Conference on Computer Science and Information Systems*, Wrocław, Poland, 9–12 September, 2012 [IEEE, Proceedings of the Federated Conference on Computer Science and Information Systems, pp. 415–421, 2012].