Available online at www.sciencedirect.com
ScienceDirect
European Journal of Operational Research 177 (2007) 733–749
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

Discrete Optimization

Solving the semi-desirable facility location problem using bi-objective particle swarm

Haluk Yapicioglu [a], Alice E. Smith [a,*], Gerry Dozier [b]

[a] Department of Industrial and Systems Engineering, Auburn University, Auburn, AL 36849, USA
[b] Department of Computer Science and Software Engineering, Auburn University, Auburn, AL 36849, USA

Received 4 April 2005; accepted 21 November 2005
Available online 9 February 2006

Abstract

In this paper, a new model for the semi-obnoxious facility location problem is introduced. The new model is composed of a weighted minisum function to represent the transportation costs and a distance-based piecewise function to represent the obnoxious effects of the facility. A single-objective particle swarm optimizer (PSO) and a bi-objective PSO are devised to solve the problem. Results are compared on a suite of test problems and show that the bi-objective PSO produces a diverse set of non-dominated solutions more efficiently than the single-objective PSO and is competitive with the best results from the literature. Computational complexity analysis estimates only a linear increase in effort with problem size.

Keywords: Multiple criteria analysis; Semi-obnoxious; Location; Evolutionary computation; Bi-objective PSO

1. Introduction

The semi-obnoxious facility location problem [5–8,17,30,37,41] has drawn much recent attention among researchers. Erkut and Neuman defined an obnoxious facility as one that generates a disservice to the people nearby while producing an intended product or service [17]. If minimization of the undesirable effects is the only concern, transportation costs to/from the facility to be located may be prohibitive. Taking this into account, Brimberg and Juel [8] introduced the term *semi-desirable facility* in 1998. They argued that the facilities cannot be classified as being purely desirable or purely obnoxious. Garbage dump sites, airports and power plants are typical semi-desirable facilities [8,17]. However, as noted in [17], the first paper that

* Corresponding author. Tel.: +1 334 844 4340; fax: +1 334 844 1381.
E-mail address: smithae@auburn.edu (A.E. Smith).

0377-2217/$ - see front matter
doi:10.1016/j.ejor.2005.11.020

recorded the concept of partially noxious facilities dates back to 1975. The terms semi-desirable and semi-obnoxious have been used interchangeably to refer to facilities that fit the definition of Brimberg and Juel [8]. With the growing environmental awareness and the increasing number of semi-desirable facilities, the necessity of addressing the location of such facilities has become more important.

Although the term semi-desirable facility dates back to 1975, most studies to date have focused on facilities that are either desirable or obnoxious. Erkut and Neumann [17] cited the need for the use of multi-objective models due to the conflicting objectives of the problem. They also reported three studies [20,10,38] which employ multi-objective models for the obnoxious facility location problem.

In this paper, a new bi-objective model and a heuristic-based solution approach are proposed. The remainder of this paper is as follows. In Section 2, a review of the bi-objective semi-desirable facility location problem (SDFLP) is given. Next, the model is introduced. Then the heuristic-based solution procedure, particle swarm optimization (PSO), and the extension of PSO to bi-objective models are presented. Computational experience of the bi-criteria PSO along with a single objective PSO on four test problems from the literature are presented. Finally, conclusions are in the last section.

2. Previously proposed bi-objective models

The use of bi-objective models for the SDFLP can be classified into two categories. The first is models which represent the transportation costs and the environmental impact in the same objective function, usually a linear combination. In the second category, the models have decoupled functions for each objective, and the solution methodologies aim at obtaining the set of efficient points and/or the Pareto front [29,30,41].

One of the models in the first category is by Romero-Morales et al. [37]. In their model, the environmental (e.g., social) cost is represented as a non-increasing convex function of the distance and the transportation cost is represented as a non-decreasing Lipschitz continuous function. The objective function is the summation of these two functions over the population centers. Romero-Morales et al. [37] proved that their objective function is non-convex and used the branch and bound procedure BSSS (Big Square Small Square) with an improvement in its bounding process to solve the problem.

Brimberg and Juel's [8] model for locating a semi-desirable facility is based on a bi-criteria model. It uses two objective functions with their weights (coefficients) summing to one. The first is the minisum criterion and the second minimizes the weighted sum of Euclidean distances raised to a negative power. To solve this model, Brimberg and Juel [8] propose a heuristic based on a branch and bound algorithm called GBSSS [36] which obtains representative points efficiently. These representative points are then used to construct the search trajectory by using a set of differential equations obtained from the objective function. In early work by Yapicioglu et al. [42], the same problem was solved by particle swarm optimization (PSO). The PSO consistently gives better results than [8] for the given values of weights of the objective functions.

The Brimberg–Juel model [8] was also studied by Skriver and Andersen [41]. The latter authors used the same objective functions as in [8] but instead of combining the two objective functions, the authors considered the model bi-objective. Their solution approach for the planar case was an approximation algorithm based upon GBSSS. They also extended the model to the network case and both problems were solved for an example airport location problem.

Brimberg and Juel [7] proposed another solution approach to the minisum maximin bi-criteria facility location problem. In this approach, the bi-criteria model is represented as a single-objective problem (minisum criterion) with a constraint (maximin $\geqslant r$; where r represents the minimum acceptable proximity for the semi-desirable facility to a population center). To solve the problem, they first obtain the optimal solution for the minisum criterion without taking the constraint into consideration. For this solution, a value of r ($\underline{r}$) is calculated. By increasing the value of r from $\underline{r}$ toward infinity, they obtain a set of efficient points for

the problem. The main drawback of this approach is that the procedure obtains the Pareto optimal points one at a time.

In another study [6], Brimberg and Juel consider the minisum criterion with convex forbidden regions constructed around each population center. They solved the model for Euclidean travel distances by using Lagrangean relaxation. For rectilinear travel distances, they propose another approach where the solution space is divided into cells by horizontal and vertical lines that pass through fixed points. Then, the optimal solution to the problem is proved to occur either on the boundary segments generated by the fixed points or on the intersection points of such boundary segments.

The bi-objective model consisting of minisum and maximin objectives was also studied by Melachrinoudis [29] and Melachrinoudis and Xanthopulos [30]. In [29], rectilinear were used as the distance metric, whereas in [30] the distance metric was Euclidean. The solution approach used in [29] was a variation of the Fourier–Motzkin Elimination Method (FMEM). To use FMEM the solution space is decomposed into K rectangular subregions where $K \leqslant (n+1)^2$, n being the number of fixed points. Then, for each of these subregions, bi-criteria linear subproblems are generated and solved using FMEM. Finally, the solutions of the subproblems are combined. Although Melachrinoudis' method is efficient in determining the Pareto front, the number of subproblems increases quadratically as the number of population centers increase. Additionally, the subproblem searches generate many points that are not optimal for the overall problem. In [30] the maximin–minisum problem with the Euclidean norm is considered. The solution procedure is based on the Karush–Kuhn–Tucker [4] conditions and Voronoi diagrams [32].

3. A new model for the SDFLP

In this section, a new bi-criteria model for the SDFLP is presented. The first criterion is the weighted minisum criterion, representing the transportation costs to be minimized. The second criterion is a piecewise non-increasing function that reflects the undesirable effects of the facility.

3.1. Minimization of transportation costs

The minimization of transportation costs has the following form:

$$\text{Minimize} \quad W_1(\mathbf{x}) = \sum_{i=1}^{n} w_i d(\mathbf{x}, \mathbf{a}_i) \tag{1}$$

The total weighted distance to the facility to be located is minimized where

i is the index for fixed points $i = 1, 2, \ldots, n$.
w_i is the unit weight associated with transportation costs for fixed point $\mathbf{a}_i$.
$d(\cdot)$ is the distance (normally Euclidean or rectilinear) between point $\mathbf{a}_i$ and the facility located at point $\mathbf{x}$.

3.2. Minimization of undesirable effects

When the location of the facility is too close to one of the fixed points, small increases in the distance do not decrease the obnoxious effect(s) of the facility very much. Thus, for up to a certain distance, the obnoxious effects of the facility can be considered constant. Similarly, when the location of the facility is distant from a population center, small decreases in the distance do not increase the obnoxious effect(s). Beyond a certain distance, the obnoxious effects of the facility can be assumed non-existent. See Fig. 1 for clarification. Using these ideas, the objective function for obnoxiousness is defined as follows:

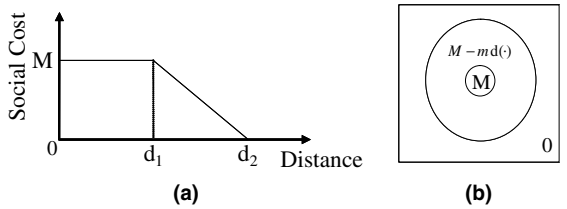

Fig. 1. Representation of the criterion in objective space (a) and decision (variable) space (b).

$$W_2(\mathbf{x}) = \sum_{i=1}^{n} \begin{cases} M & \text{if } d(\mathbf{x}, \mathbf{a}_i) \leqslant d_1, \\ M - m(d(\mathbf{x}, \mathbf{a}_i)) & \text{if } d_1 < d(\mathbf{x}, \mathbf{a}_i) < d_2, \\ 0 & \text{if } d_2 \leqslant d(\mathbf{x}, \mathbf{a}_i), \end{cases} \tag{2}$$

where i is the index for fixed points $i = 1, 2, \ldots, n$. $d(\cdot)$ is the distance between fixed point $\mathbf{a}_i$ and the facility located at point $\mathbf{x}$.

The criterion in (2) provides more flexibility than the maximin criterion. In the model above, M and m can be set with respect to the specifics of the problem at hand as can the distance thresholds d_1 and d_2.

4. Particle swarm optimization (PSO)

Particle swarm optimization (PSO) is an Evolutionary Computation (EC) method developed by James Kennedy and Russell Eberhart to solve continuous non-linear functions [1,26–28,39]. PSO was inspired by the flocking/schooling of birds and fish, and simulated evolution [3,19,21,22]. Since its conception in 1995 [2,25–28,33,39,40], PSO has been noted for two main features: optimization via social evolution [26–28] and its relative ease of use. In PSO, members of the swarm can be considered agents (i.e., particles) searching through the solution space. A particle is composed of three vectors and two fitness values. The *x-vector* records the current position of the particle in the search space. The *p-vector* records the location of the best position found so far by this particle and the *v-vector* contains the direction in which the particle will travel. Both the x-vector and the p-vector have associated fitness values.

At initialization, the x-vector and v-vector for each particle are randomly generated within a user defined range. For each particle, the x-fitness is calculated and the p-vector and p-fitness are set equal to the x-vector and x-fitness, respectively. In the next step, particles are updated. The v-vector is updated first and then added to x-vector according to the formulas given below

$$v_{id} = v_{id} + \varphi_1 \text{rnd}()(p_{id} - x_{id}) + \varphi_2 \text{rnd}()(p_{gd} - x_{id}), \tag{3}$$

$$x_{id} = x_{id} + v_{id}, \tag{4}$$

where i is the ith particle, φ_1 and φ_2 are learning rates [1] governing the cognition (particle's own experience) and social components (communication with neighborhood), respectively, g is the index of the particle with the best p-fitness, d is the dth dimension of the v-vector and rnd() is a uniform random number in the interval $[0, 1]$. After the x-vector is updated, the new x-fitness value is calculated. If the new x-fitness is better than the p-fitness for the particle, the p-vector is replaced with the new x-vector, and the p-fitness is set equal to the x-fitness.

In PSO, the most widely used neighborhood structures are the Ring Topology and the Star Topology. In the Ring Topology, the neighborhood size is assumed to be three; each particle can communicate with its two nearest neighbors. In the Star Topology, the neighborhood size is the entire swarm (a global neighborhood).

4.1. Stability and explosion in PSO

Due to the random components in Eq. (3), PSO is susceptible to an explosion of particles' velocities and positional coordinates [11]. One way to overcome this problem is to set upper and lower bounds for the v-vector [11]. There are two other mechanisms that can be used to control explosion of velocities and provide stability. These are inertia, proposed by Shi and Eberhart [39], and the constriction coefficient; proposed by Clerc [11].

Using inertia, the velocity update formula is

$$v_{id} = wv_{id} + \varphi_1 \mathrm{rnd}()(p_{id} - x_{id}) + \varphi_2 \mathrm{rnd}()(p_{gd} - x_{id}), \tag{5}$$

where w is the inertia whose value decreases linearly as the number of iterations of the algorithm increases.

Using the constriction coefficient, the v-vector is updated by

$$v_{id} = K(v_{id} + \varphi_1 \mathrm{rnd}()(p_{id} - x_{id}) + \varphi_2 \mathrm{rnd}()(p_{gd} - x_{id})), \tag{6}$$

$$\text{where } K = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|} \quad \text{and} \quad \varphi = \varphi_1 + \varphi_2 > 4. \tag{7}$$

Both of these measures have been found to be useful in improving the performance of PSO [11,16,39].

4.2. Multi-objective PSO

In multi-objective optimization, the aim is to simultaneously optimize a set of (sometimes conflicting) objective functions. A multi-objective model has the form [14]

$$\begin{aligned} &\text{Minimize/Maximize} && f_m(\mathbf{x}), \quad m = 1, 2, \ldots, M \\ &\text{Subject to} && g_j(\mathbf{x}) \geqslant 0, \quad j = 1, 2, \ldots, J, \\ & && h_k(\mathbf{x}) = 0, \quad k = 1, 2, \ldots, K, \\ & && x_i^{(L)} \leqslant x_i \leqslant x_i^{(U)}, \quad i = 1, 2, \ldots, n, \end{aligned} \tag{8}$$

where M is the number of objective functions, subject to J inequality and K equality constraints where n decision variables denoted by $\mathbf{x}$ are bounded from below and above by $\mathbf{x}^{(L)}$ and $\mathbf{x}^{(U)}$, respectively. The general form of the multi-objective models presented above requires the decision maker to define a method to compare alternative solutions against each other. Two of the most widely used methods are Pareto preference and value preference [43]. In Pareto preference, a solution, $\mathbf{x}_{(2)}$, is dominated by solution, $\mathbf{x}_{(1)}$, if $\mathbf{x}_{(1)}$ is not worse than $\mathbf{x}_{(2)}$ in all objectives, and for at least one of the objectives, $\mathbf{x}_{(1)}$ is strictly better than $\mathbf{x}_{(2)}$ [18]. Without loss of generality, these conditions can be expressed as follows for the case where all of the objective functions are to be minimized:

$$\begin{aligned} &f_m(\mathbf{x}_{(1)}) \leqslant f_m(\mathbf{x}_{(2)}) \quad \text{for } \forall m = 1, 2, \ldots, M \text{ and} \\ &f_m(\mathbf{x}_{(1)}) < f_m(\mathbf{x}_{(2)}) \quad \text{for some } m. \end{aligned} \tag{9}$$

In value preference, $\mathbf{x}_{(2)}$ is worse than a solution, $\mathbf{x}_{(1)}$, if for function $g(\mathbf{x})$, defined over $f_m(\mathbf{x})$, $g(\mathbf{x}_{(1)}) \leqslant g(\mathbf{x}_{(2)})$. In general, $g(\mathbf{x})$ is represented by the linear combination of $f_m(\mathbf{x})$, $m = 1, 2, \ldots, M$.

The set of solutions where no solution is dominated by any other solution is the set of non-dominated solutions [14]. The non-dominated set of the entire feasible search space is the Pareto optimal set or Pareto front [14]. The efficient set is the set of solutions of the Pareto front in variable space, whereas the Pareto front is defined in objective function space. Identification of the efficient set for a given facility location problem is especially important because the efficient set are the location alternatives for the facility.

4.2.1. Previous work in multi-objective PSO

The first attempt to modify PSO to handle multi-objective optimization problems was by Moore and Chapman [31]. They modify the *p*-vector of the particles so that each particle keeps track of all non-dominated solutions (using Pareto preference) experienced by itself. In another study, Coello Coello and Lechuga [12] propose the use of an external repository to keep non-dominated solutions. If the repository is full and a newly discovered non-dominated solution lies in a less crowded area of the Pareto front, it replaces a non-dominated solution that lies in a more crowded area [12].

Hu and Eberhart [23] introduce the use of a dynamic neighborhood to the PSO to cope with multi-objective optimization problems. After each update of the swarm, particles update their neighborhoods. The new neighborhoods for the particles are determined in terms of the proximity calculated by using one of the objective functions'. Then, the particles in each neighborhood try to optimize the other objective function to move towards the Pareto front. Hu et al. [24] extend this work by using an external repository to store the current set of Pareto optimal solutions.

Parsopoulos and Vrahatis [35] propose a weighted aggregation method with three different variants to calculate the weights. Another study by Parsopoulos et al. [34] suggests using multiple swarms where the number of swarms is equal to the number of objective functions. Each swarm searches one objective function and the best solution found by a swarm is fed to another swarm to direct the search of the particles of that swarm.

Fieldsend and Singh [18] propose a new data structure to cope with the shortcomings of using a constant size repository. The "dominated tree" structure represents non-dominated solutions compactly. They also studied a stochastic component to increase the effectiveness of the PSO algorithm in multi-objective optimization [18].

A recent study on multi-objective PSO is by Coello Coello et al. [13]. An external repository and a mutation operator are employed. The external repository is composed of two parts: the archive controller and the grid. The archive controller is where non-dominated solutions are kept and newly found solutions are checked against. The grid is used to ensure a uniform distribution of non-dominated solutions along the Pareto front. A mutation operator encourages full exploration of the solution space.

4.2.2. Proposed bi-objective PSO

In the bi-objective PSO in this paper, a repository for non-dominated solutions and a routine by Deb [14] to update the Pareto front are used. The steps of the bi-PSO are:

```
Procedure Bi-objective PSO{
  t = 0;
  Initialize Swarm(t); //Swarm at tth cycle
  Evaluate the particles of the Swarm(t);
  Initialize Pareto_Front (t); //Repository at tth cycle
  t++;
  While (Not Done) {
    for each particle i ∈ Swarm(t) {
      Select a member from Pareto_Front(t) randomly;
      Update (i, Pareto_Front(t));
      Evaluate particle i;
    }
  Update Pareto_Front(t);
  t = t + 1;
  }
}
```

To update the velocity vector, for each member of the swarm a different member from the current repository is selected. Hence, the velocity update formula is modified to the following:

$$g = \text{int}(\text{rnd}()|\text{Pareto\_Front}|), \tag{10}$$

$$v_{id} = K(v_{id} + \varphi_1 \text{rnd}()(p_{id} - x_{id}) + \varphi_2 \text{rnd}()(p_{gd} - x_{id})). \tag{11}$$

The equation given in (10) allows the algorithm to select a member of the repository randomly to update the velocity of particle i. Note that the only change in the velocity update formula is the selection of g.

Three variants of the bi-PSO are devised:

- bi-PSO with a constriction coefficient: (bi-PSO$_{cc}$),
- bi-PSO without a constriction coefficient: (bi-PSO$_{wocc}$), (this is the case where $K = 1$),
- Hybrid bi-PSO where half of the swarm updates its velocity vector using a constriction coefficient and the other half does not use a constriction coefficient (bi-PSO$_{hyb}$).

The development of these three variants is based on results recently published in [15]. When using the constriction coefficient, the PSO tends to explore the inner regions of the search space more thoroughly, but it does not perform as well at regions near the boundaries of the solution space. Without a constriction coefficient the movement of the particles might be unstable, however the PSO searches regions near the boundaries of the search space better. The bi-PSO$_{hyb}$ aims to balance these.

5. Computational experience

A single-objective PSO (uni-PSO) from [42] and the bi-PSOs from this paper solved the models of [29,30,41] using four test problems from the literature. In [29,30], a weighted minisum function minimizes the transportation costs and a maximin function minimizes the undesirable effects of the facility over the fixed points. In [41], the formulation of transportation costs is the same as [29,30], but to minimize the undesirable effects the weighted sum of Euclidean distances raised to the power of -2 is used. Test problem 1 is adopted from [8], whereas test problem 2 and test problem 3 are adopted from [30] and [29], respectively. Problem 4 is taken from [41]. All problem data are summarized in Table 1, except Problem 4 (which is available in [41]).

The parameters of the model for test problem 1 were set as in Table 2.

The parameters and the distance metrics for the remaining problems were set as the same as reported in their respective references.

Table 1
Data for the first three test problems

	i	1	2	3	4	5	6	7
Problem 1	$\mathbf{a}_i$	(5, 20)	(18, 8)	(22, 16)	(14, 17)	(7, 2)	(5,15)	(12, 4)
	w_1	5	7	2	3	6	1	5
	w_2	1	1	1	1	1	1	1
Problem 2	$\mathbf{a}_i$	(4, 4)	(8, 7)	(11, 10)	(13, 4)			
	w_1	2	4	3	2			
	w_2	1	1	1	1			
Problem 3	$\mathbf{a}_i$	(1, 3)	(4, 5)	(6, 1)	(6, 7)	(8, 5)		
	w_1	3	2	3	1	2		
	w_2	1	1	1	1	1		

Table 2
Parameter settings for Problem 1

Parameter	Value
M	200
m	1
d_1	10
d_2	30

5.1. Uni-PSO (value preference)

In the uni-PSO a weighted sum objective function, $g(\mathbf{x}) = (1 - \lambda)f_1(\mathbf{x}) + \lambda f_2(\mathbf{x})$ where $0 \leqslant \lambda \leqslant 1$ with $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ representing transportation costs and obnoxious effects, respectively, was used. To obtain a basis of comparison with the results of the bi-objective PSO, the value of λ was changed from 0 to 1 in increments of 0.01. Thus, for each test problem, there are 101 representative points of the Pareto front. The parameters of the uni-PSO were $\varphi_1 = \varphi_2 = 2.05$ (robust values identified in [9]) with a constriction coefficient (Eq. (7)) and a global neighborhood.

5.2. The bi-objective PSO (Pareto preference)

As with the uni-PSO, $\varphi_1 = \varphi_2 = 2.05$ and a constriction coefficient was used. For each problem, the bi-objective PSO was run with 500 particles for 2000 iterations. Although the swarm size is larger than normal, smaller swarm sizes did not generate a Pareto front with an adequate coverage for the problems considered. For each case, a single run of the bi-objective PSO was performed and the external repository at the end of the run was declared the Pareto set.

5.3. Results

For Problems 2 and 3 all three variants of bi-objective PSO were initiated with two non-dominated solutions; the best solutions of each objective function found by using global neighborhood uni-PSO with a constriction coefficient. The initialized (also termed "seeded") bi-objective PSO identified more non-dominated solutions than the unseeded case. Although bi-PSO_{cc} discovers more non-dominated solutions for all test problems, bi-PSO_{wocc} and bi-PSO_{hyb} identified a wider Pareto front. To balance diversity and quantity, bi-PSO_{hyb} is chosen as the subject of further result comparisons.

As shown in Figs. 2, 3 and 5, the uni-PSO performs poorly in generating the Pareto front and efficient sets for Problems 1.1, 1.2 and 3. For Problems 2 and 4, the uni-PSO generates the lower left side and the upper left side of the Pareto fronts, respectively (Figs. 4 and 6). But the uni-PSO fails to locate three regions in the former and two regions in the latter even though the search space considered is much small than in the two objective case (recall that one objective function value is set while the other is searched).

In Fig. 2, the efficient set and the Pareto front found by bi-PSO_{hyb} for Problem 1.1 are shown. Discontinuities in the Pareto front are due to the piecewise structure of the second objective function.

For Problem 1.2, the efficient set and the Pareto front found by bi-PSO_{hyb} are given in Fig. 3. One can see in Table 3, that when the distance changes from Euclidean to rectilinear, the number of non-dominated solutions decreases by almost 35%. This is because the rectilinear distance between two points is always greater than or equal to the Euclidean distance. As the facility is moved from the fixed points, the rectilinear distance function decreases faster, so the threshold values for the social cost function are encountered earlier. To further decrease social cost, another threshold has to be reached. However, the transportation costs increase more with the rectilinear distance function. As a result, the drop in the social costs cannot accom-

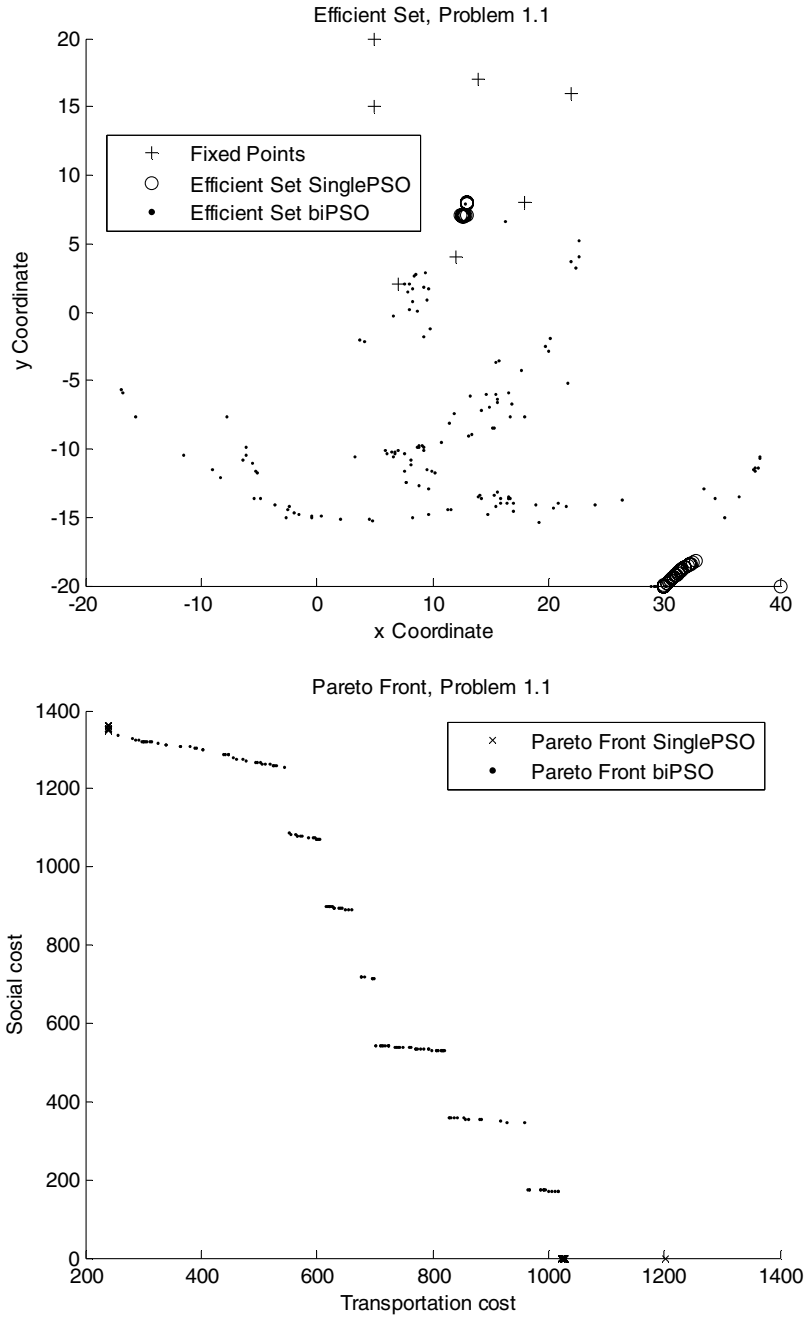

Fig. 2. Problem 1.1. Efficient set and Pareto front.

modate the increase in transportation cost for certain regions. A closer examination of the efficient sets for rectilinear distances reveals that the bi-PSO_{hyb} does not find any non-dominated solutions in the region $-15 \leqslant x \leqslant 5$ (see Fig. 3).

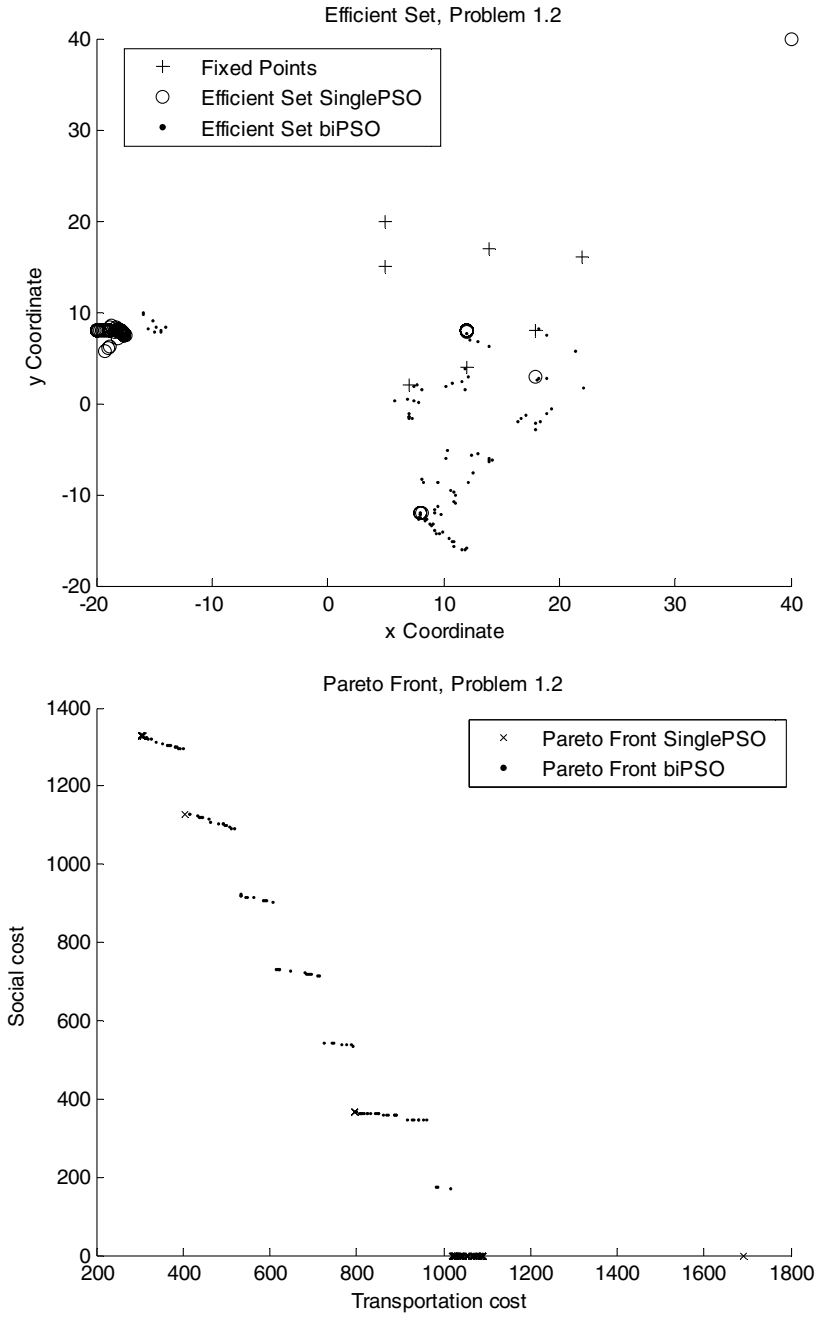

Fig. 3. Problem 1.2. Efficient set and Pareto front.

For Problem 2, the efficient set and the Pareto front found by bi-PSO$_{hyb}$ are given in Fig. 4. The bi-PSO-$_{hyb}$ obtains almost the same Pareto front as found in [30] where the solution method is exact.

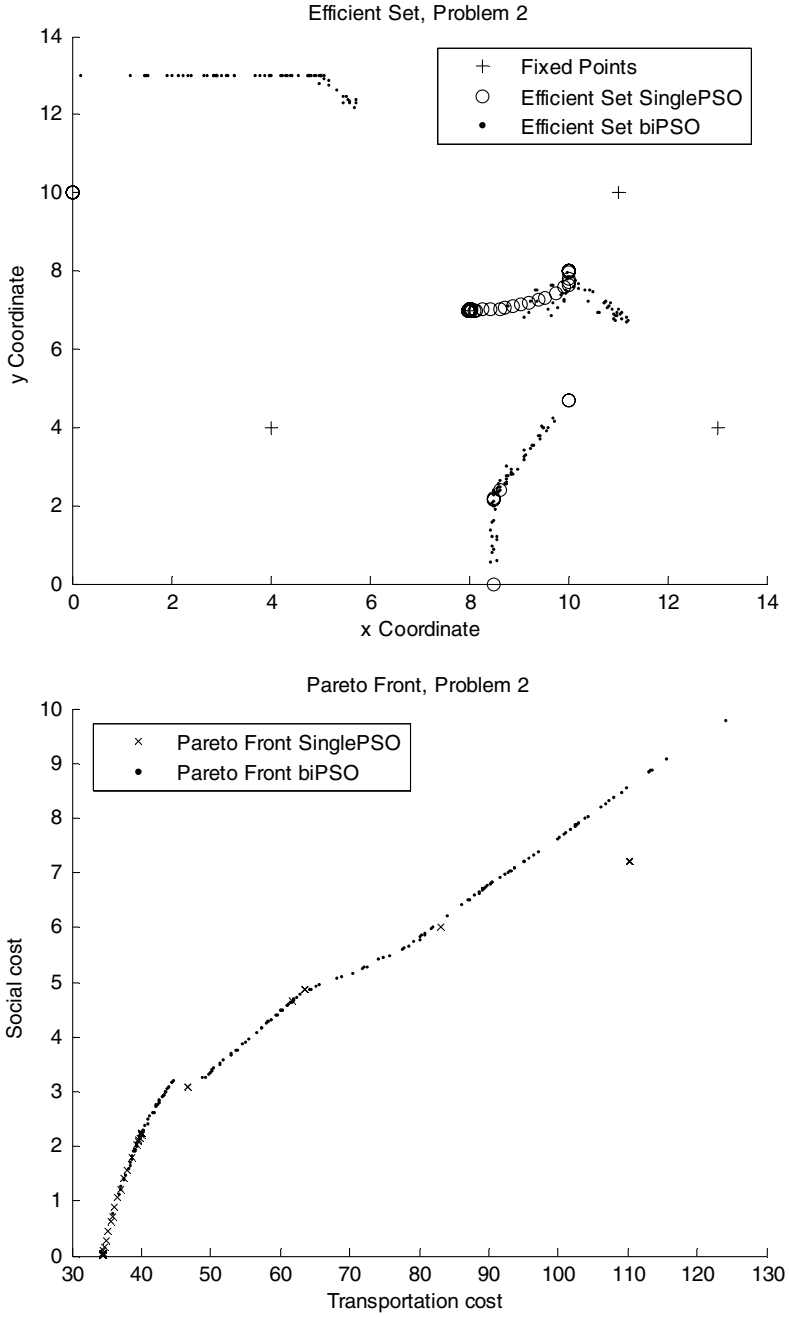

Fig. 4. Problem 2. Efficient set and Pareto front.

The minisum–maximin bi-objective model with rectilinear distances is the third example problem. The performance of bi-PSO_{hyb} is comparable with the results of [29]. To provide a basis of comparison, the efficient set and the Pareto front discovered by bi-PSO_{hyb} is provided in Fig. 5.

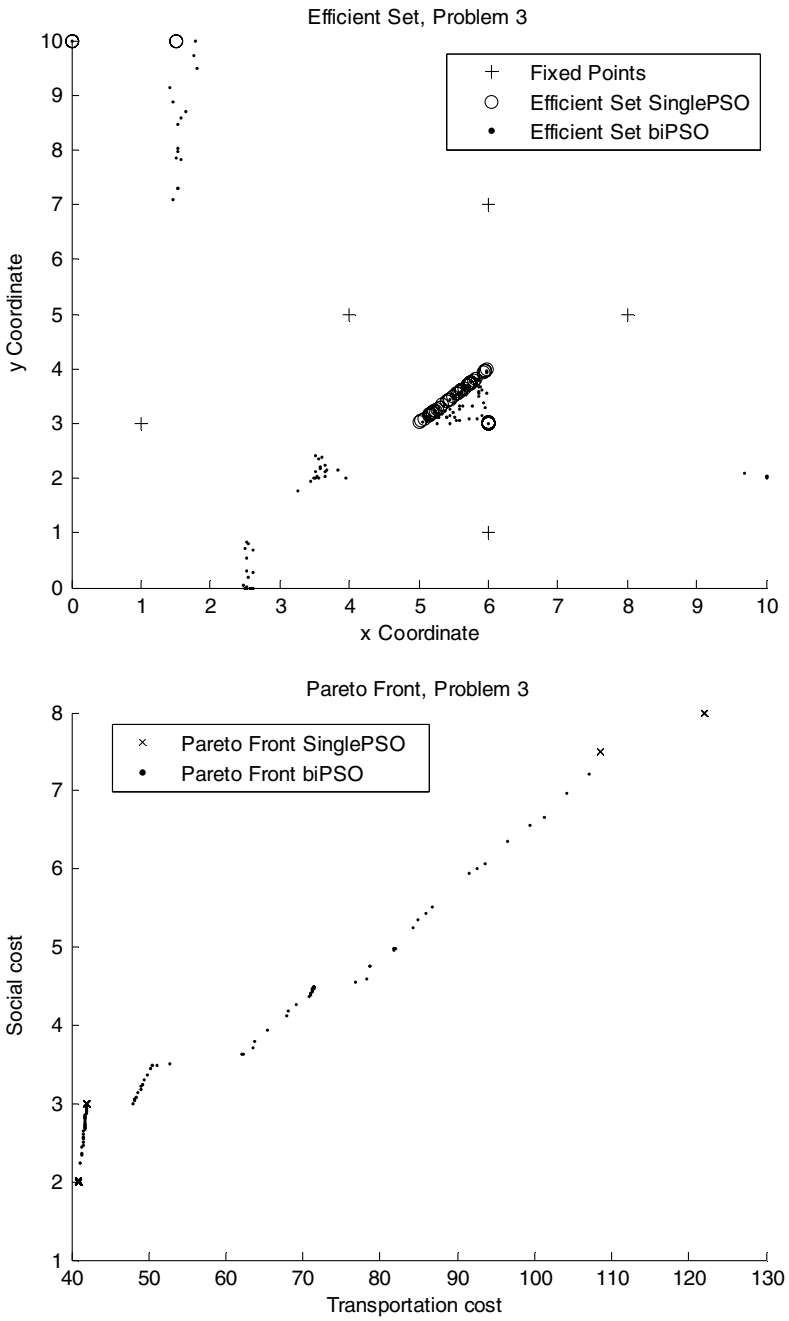

Fig. 5. Problem 3. Efficient set and Pareto front.

The last example problem is different from the previous ones both in terms of the objective function representing the obnoxious effects and the size of the search space. The bi-PSO_{hyb} finds almost the same efficient set as that of [41]. For this problem the upper left side of the Pareto front is more extensively

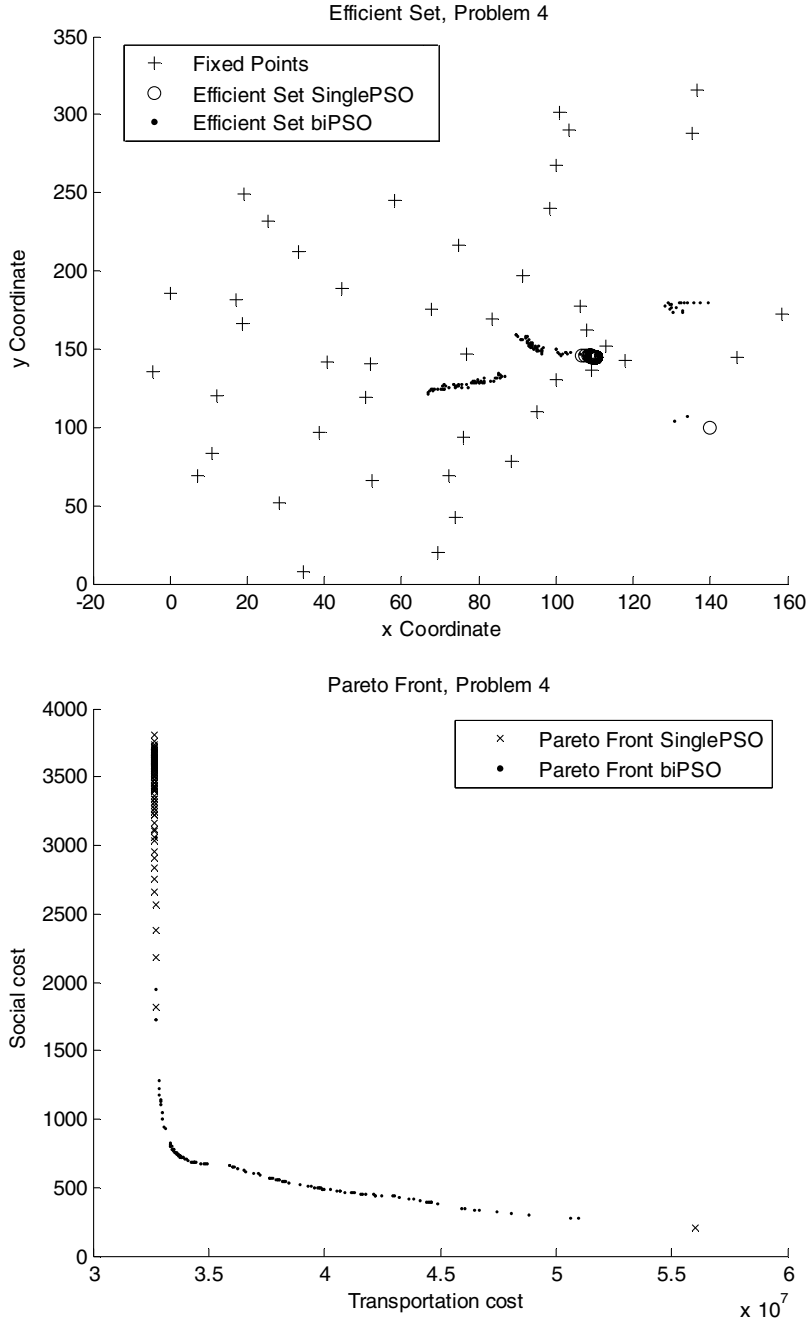

Fig. 6. Problem 4. Efficient set and Pareto front.

discovered by the uni-PSO, but all of the non-dominated solutions correspond to the same region of the efficient set. However, there are two more regions of the solution space in the efficient set and these regions are discovered by bi-PSO_{hyb}.

Table 3
Number of non-dominated solutions found

	Problem 1.1	Problem 1.2	Problem 2	Problem 3	Problem 4
Bi-PSO$_{cc}$	148	95	129	157	148
Bi-PSO$_{wocc}$	126	79	97	142	114
Bi-PSO$_{hyb}$	143	94	114	142	128

5.4. Computational effort

In all cases tested the bi-PSO$_{hyb}$ provides comparable or better results than those in the literature. Clearly, the bi-PSO$_{hyb}$ is a viable approach to the bi-criteria SDFLP. Now, computational effort must be considered. Table 4 summarizes the computational complexities of the solution methods cited in this paper. Although the bi-PSO$_{hyb}$ does not guarantee generating the exact Pareto front or efficient set, its order of complexity is significantly less than the methods reported in [6,7,29,30]. (A comparison of the bi-PSO$_{hyb}$ with Skriver and Andersen's method [41] could not be done because no information on the algorithm's computational complexity was reported.)

In Table 5 the computational times required for bi-PSOs are summarized. The algorithms were coded with Microsoft Visual C++ 6.0 and run on a PC with a Pentium-IV 3 GHz processor and 512 MB of RAM running a Windows XP Operating System. Even for the large problem (45 fixed points), generating the Pareto front and the efficient set does not take more than a minute. Since none of the papers cited provide information on computational time, the comparison among the bi-PSOs only is made. For all three variants, computation time does not change significantly, however, as the number of fixed points increases, computational time required does increase. The computational time requirement changes linearly with respect to the number of fixed points without being affected by the type of objective functions or distance metric used (see Fig. 7). The trend line fitted to the plot has an R^2 value of 95%. In order to strengthen the

Table 4
Comparison of algorithms

Algorithm	Distance metric	Nature of the solution approach	Order of complexity	Comparison problem
Bi-objective PSO (this paper)	Euclidean or rectilinear	Approximate	$O(n)$	Problem 1.1 Problem 1.2
Melachrinoudis [29]	Rectilinear	Exact	$O(n^2)$	Problem 2
Melachrinoudis and Xanthopulos [30]	Euclidean	Exact	$O(m\log^2 n + n\log n + n\log m)$	Problem 3
Skriver and Andersen [40]	Euclidean	Approximate	N/A	Problem 4
Brimberg and Juel [5]	Rectilinear	Approximate	$O(n^2)$	N/A
Brimberg and Juel [6]	Rectilinear	Exact	$O(n^2\log(n))$	N/A
Brimberg and Juel [6]	Euclidean	Exact	$O(n^3)$	N/A

Note: n is the number of fixed points and m is the number of the edges.

Table 5
Average computational time to obtain efficient sets and Pareto fronts (in CPU seconds)

	Problem 1.1	Problem 1.2	Problem 2	Problem 3	Problem 4
bi-PSO$_{cc}$	20.633	16.514	13.276	12.104	54.976
bi-PSO$_{wocc}$	19.853	16.925	11.688	11.420	53.268
bi-PSO$_{hyb}$	20.181	16.702	12.252	11.015	52.533

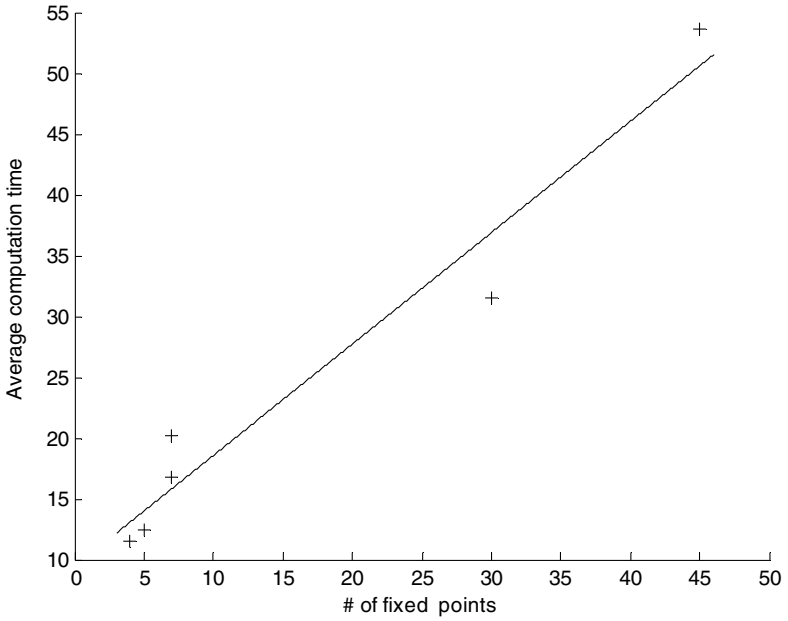

Fig. 7. Average computation time vs. number of fixed points.

conclusion drawn from the computational times, another problem with 30 fixed points with the Euclidean distance metric was devised and tested.

6. Conclusion

This paper puts forth a new model for the SDFLP that is flexible and tractable. Three versions of a bi-PSO are devised to solve the problem. Although all are effective, to balance breadth and depth of the Pareto front, a hybrid version is recommended which includes solutions both with and without the constriction coefficient. Results on several well known test problems are compared with heuristics and exact methods from the literature and with a single objective version of the PSO. The bi-objective PSO is convincing in its computational efficiency and its ability to find numerous, well spaced non-dominated solutions. The computational complexity of the algorithm is estimated as linear with number of fixed points, making this method applicable even for very large problems.

All of the example problems and the model presented here are for the planar case. One direction for further research could be to choose distance metrics more realistically, e.g., the use of network distances for the transportation and planar distances for the obnoxiousness seem more appropriate. More realistic ways of modeling the obnoxious effects of a given facility should be investigated such as non-linear and piecewise functions. Transportation costs could be considered not as distance related but as travel time related. These alterations can readily be accommodated by the PSO framework described herein.

References

[1] P.J. Angeline, Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences, in: Evolutionary Programming VII, Lecture Notes in Computer Science, vol. 1447, Springer-Verlag, Berlin, 1998, pp. 601–610.

[2] P.J. Angeline, Using selection to improve particle swarm optimization, in: Proceedings of the 1998 IEEE International Conference on Evolutionary Computation, 1998, pp. 84–89.
[3] T. Bäck, D.B. Fogel, Z. Michalewicz (Eds.), Evolutionary computation, in: Basic Algorithms and Operators, vol. 1, 2000, Institute of Physics Pub, Bristol, Philadelphia.
[4] M.S. Bazaraa, J.J. Jarvis, H.D. Sherali, Linear Programming and Network Flows, third ed., John Wiley & Sons, Ltd., New York, 2004.
[5] O. Berman, Z. Drezner, A note on the location of an obnoxious facility on a network, European Journal of Operational Research 120 (1) (2000) 215–217.
[6] J. Brimberg, H. Juel, A minisum model with forbidden regions for locating a semi-desirable facility in the plane, Location Science 6 (1998) 109–120.
[7] J. Brimberg, H. Juel, On locating a semi-desirable facility on the continuous plane, International Transactions in Operational Research 5 (1) (1998) 59–66.
[8] J. Brimberg, H. Juel, A bi-criteria model for locating a semi-desirable facility in the plane, European Journal of Operational Research 106 (1) (1998) 144–151.
[9] A. Carlisle, G.V. Dozier, An off-the-shelf PSO, in: Proceedings of the 2001 Workshop on Particle Swarm Optimization, 2001, pp. 1–6.
[10] R.L. Church, J. Cohon, Multiobjective location analysis of regional energy facility siting problems, Report No. BNL 50567, Policy Analysis Division, Brookhaven National Lab., Upton NY, 1976.
[11] M. Clerc, J. Kennedy, The particle swarm—explosion, stability, and convergence in a multidimensional complex space, IEEE Transactions on Evolutionary Computation 6 (1) (2002) 58–73.
[12] C.A. Coello Coello, M.S. Lechuga, MOPSO: A proposal for multiple objective particle swarm optimization, in: Proceedings of the 2002 Congress on Evolutionary Computation, vol. 2, 2002, pp. 1051–1056.
[13] C.A. Coello Coello, G.T. Pulido, M.S. Lechuga, Handling multiple objectives with particle swarm optimization, IEEE Transactions on Evolutionary Computation 8 (3) (2004) 256–279.
[14] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, John Wiley & Sons, New York, 2001.
[15] G. Dozier, D. Brown, J. Hurley, K. Cain, Vulnerability analysis of AIS-based intrusion detection systems via genetic and particle swarm red teams, in: Proceedings of the 2004 Congress on Evolutionary Computation, vol. 1, 2004, pp. 111–116.
[16] R.C. Eberhart, Y. Shi, Comparing inertia weights and constriction factors in particle swarm optimization, in: Proceedings of the 2000 Congress on Evolutionary Computation, vol. 1, 2000, pp. 16–19.
[17] E. Erkut, S. Neuman, Analytical models for locating undesirable facilities, European Journal of Operational Research 40 (3) (1989) 275–291.
[18] J.E. Fieldsend, S. Singh, A multi-objective algorithm based upon particle swarm optimisation, an efficient data structure and turbulence, in: Proceedings of the 2002 UK Workshop on Computational Intelligence, 2002, pp. 37–44.
[19] L.J. Fogel, A.J. Owens, M.J. Walsh, Artificial Intelligence through Simulated Evolution, John Wiley & Sons, New York, 1966.
[20] A. Goicoechea, D.R. Hansen, L. Duckstein, Multiobjective Decision Analysis with Engineering and Business Applications, John Wiley & Sons, New York, 1982.
[21] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, New York, 1989.
[22] J.H. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, University of Michigan Press, Ann Arbor, 1975.
[23] X. Hu, R. Eberhart, Multiobjective optimization using dynamic neighborhood particle swarm optimization, in: Proceedings of the 2002 Congress on Evolutionary Computation, vol. 2, 2002, pp. 1677–1681.
[24] X. Hu, R. Eberhart, Y. Shi, Particle swarm with extended memory for multiobjective optimization, in: Proceedings of the 2003 IEEE Swarm Intelligence Symposium, Indianapolis, 2003, pp. 193–197.
[25] J. Kennedy, Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance, in: Proceedings of the 1999 Conference on Evolutionary Computation, 1999, pp. 1931–1938.
[26] J. Kennedy, The particle swarm: Social adaptation of knowledge, in: Proceedings of the 1997 International Conference on Evolutionary Computation, Indianapolis, IN, 1997, pp. 303–308.
[27] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of the 1995 IEEE International Conference on Neural Networks, IV, 1995, pp. 1942–1948.
[28] J. Kennedy, R. Eberhart, Swarm Intelligence, Morgan Kaufmann Publishers, San Francisco, 2001.
[29] E. Melachrinoudis, Bicriteria location of a semi-obnoxious facility, Computers and Industrial Engineering 37 (3) (1999) 581–593.
[30] E. Melachrinoudis, Z. Xanthopulos, Semi-obnoxious single facility location in Euclidean space, Computers and Operations Research 30 (14) (2003) 2191–2209.
[31] J. Moore, R. Chapman, Application of particle swarm to multiobjective optimization, Department of Computer Science and Software Engineering, Auburn University, 1999.
[32] A. Okabe, B. Boots, K. Sugihara, Spatial Tessellations, Concepts and Applications of Voronoi Diagrams, John Wiley & Sons, Chichester, 1992.

[33] E. Ozcan, C.K. Mohan, Particle swarm optimization: Surfing the waves, in: Proceedings of the IEEE 1999 Congress on Evolutionary Computation, 1999, pp. 1939–1944.
[34] K.E. Parsopoulos, D.K. Tasoulis, M.N. Vrahatis, Multiobjective optimization using parallel vector evaluated particle swarm optimization, in: Proceedings of the IASTED International Conference on Artificial Intelligence and Applications, vol. 2, 2004, pp. 823–828.
[35] K.E. Parsopoulos, M.N. Vrahatis, Particle swarm optimization method in multiobjective problems, in: Proceedings of the 2002 ACM Symposium on Applied Computing, 2002, pp. 603–607.
[36] F. Plastria, GBSSS: The generalized big square small square method for planar single-facility location, European Journal of Operational Research 62 (2) (1992) 163–174.
[37] D. Romero-Morales, E. Carrizosa, E. Conde, Semi-obnoxious location models: A global optimization approach, European Journal of Operational Research 102 (2) (1997) 295–301.
[38] R.K. Sarin, Ranking of multiattribute alternatives with an application to coal power siting, in: G. Fandel, T. Gal (Eds.), Multiple Criteria Decision Making: Theory and Applications, Springer-Verlag, Berlin, 1980.
[39] Y. Shi, R.C. Eberhart, A modified particle swarm optimizer, in: Proceedings of the 1998 IEEE International Conference on Evolutionary Computation, 1998, pp. 69–73.
[40] Y. Shi, R.C. Eberhart, Parameter selection in particle swarm optimization, in: Proceedings of 7th Annual Conference on Evolutionary Programming, 1998, pp. 591–600.
[41] A.J.V. Skriver, K.A. Andersen, The bicriterion semi-obnoxious location problem (BSLP) solved by an ε-approximation, European Journal of Operational Research 146 (3) (2003) 517–528.
[42] H. Yapicioglu, G. Dozier, A.E. Smith, Bi-criteria model for locating a semi-desirable facility on a plane using particle swarm optimization, in: Proceedings of the 2004 Congress on Evolutionary Computation, 2004, pp. 2328–2334.
[43] P.L. Yu, Multiple criteria decision making: Five basic concepts, in: G.L. Nemhauser, A.H.G. Rinnooy Kan, M.J. Todd (Eds.), Handbooks in Operations Research and Management Science, Optimization, Elsevier Science Publishers, Amsterdam, 1989, pp. 663–699.