OR Spectrum 28:417–436 (2006)
DOI 10.1007/s00291-005-0010-7

REGULAR ARTICLE

Zvi Drezner

Finding a cluster of points and the grey pattern quadratic assignment problem

Published online: 21 December 2005

Abstract In this paper we propose a model which aims at selecting a tight cluster from a set of points. The same formulation applies also to the grey pattern problem where the objective is to find a set of black dots in a rectangular grid with a given density so that the dots are spread as evenly as possible. A branch and bound algorithm and five heuristic approaches are proposed. Computational results demonstrate the efficiency of these approaches. Seven grey pattern problems are solved to optimality and for eight additional grey pattern problems the best known solution is improved. The cluster problem on a network is solved for 40 problems with the number of points ranging between 100 and 900 and the size of the cluster ranging between 5 and 200. Twenty one problems were solved optimally and the remaining 19 problems were heuristically solved in a very short computer time with excellent results.

Keywords Quadratic assignment problem · Metaheuristics · Cluster · Grey pattern

1 Introduction

Consider n objects (such as points in the plane or nodes of a network) with a given distance between every pair of points. We wish to find a cluster of m points which minimizes the total distance between all pairs of points in the cluster. This cluster can be interpreted as the "tightest" cluster of m points. This is similar to the one facility version of the max-cover problem (for a network or discrete formulation, see Daskin 1995; Current et al. 2002; for planar models, see Drezner 1981 for one facility, and Watson-Gandy 1982 for several facilities) where we wish to find the location of several facilities which cover the maximum number of points within a given distance.

Z. Drezner
College of Business and Economics, California State University-Fullerton,
Fullerton, CA 92834, USA
E-mail: zdrezner@fullerton.edu

Examples of applications include the selection of a group of m people out of n available people. The distance between a pair of persons is a measure of compatibility or a measure of being able to work together. The ideal group will have the most compatibility among the group members and the least potential for conflicts. Another application is selecting a subset of m points out of a given set of n points in the plane to be connected by links. We wish to minimize the total length of all links. The grey pattern problems in the context of the Quadratic Assignment Problem (Taillard 1995) can be formulated in an identical way. The grey pattern problem (Taillard 1995) is based on a rectangle of dimensions n_1 by n_2. A grey pattern of m black points is selected from the $n = n_1 \times n_2$ points in the rectangle while the rest of the points remain white. This forms a "grey pattern" of density m/n . The objective is to have a grey pattern where the black points are distributed as uniformly as possible. This objective is achieved by defining a distance between pairs of points according to some rule. For more details see Taillard (1995).

All computational experiments were performed on a 2.8 GHz Pentium IV desktop computer. Programs were coded in FORTRAN and compiled by Microsoft FORTRAN PowerStation 4.0.

2 Formulation

Let

n be the number of points,
m be the number of points to be selected for the cluster,
d_{ij} be the distance between points i and j ($d_{ij} = d_{ji}$, $d_{(ii)} = 0$).

Let M of cardinality m be the subset of selected points. The objectivefunction is to minimize

$$F(M) = \sum_{i \in M} \sum_{j \in M} d_{ij} \tag{1}$$

This problem can be formulated as a quadratic assignment problem (Rendl 2002). The QAP is a combinatorial optimization problem stated for the first time by Koopmans and Beckmann (1957). The problem is defined as follows. A set of n possible sites are given and n facilities are to be located on these sites, one facility at a site. Let c_{ij} be the cost per unit distance between facilities i and j and d_{ij} be the distance between sites i and j. The cost f to be minimized over all possible permutations, calculated for an assignment of facility i to site $p(i)$ for $i = 1, \ldots n$, is:

$$f = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} d_{p(i)p(j)} \tag{2}$$

The quadratic assignment problem is a very difficult optimization problem and only recently problems of up to $n = 36$ points were solved optimally (see Drezner et al. 2005).

We define the weight matrix $\{c_{ij}\}$ as $c_{ij}=1$ for $i, j\leq m$ and $c_{ij}=0$ otherwise, and the distance matrix is defined by the given distances $\{d_{ij}\}$. Every permutation of the points defines the selected group as the first m points of the permutation and the value of the objective function is the sum of all the distances among the selected group members. This is the formulation used by Taillard (1995) and two problems of this type of grey pattern are available at QAPLIB http://www.seas.upenn.edu/qaplib) and are called Tai64c and Tai256c. Tai64c is a grey pattern problem in a square of 8 by 8 points ($n=64$) and $m=13$ black points. Tai256c is a grey pattern problem in a square of dimensions 16 by 16 ($n=256$) and $m=92$ black points. Taillard and Gambardella (1997) define 126 grey pattern problems similar to Tai256c for $n=256$ and $3\leq m\leq 128$. Since this quadratic assignment formulation has a special structure, it is easier to solve as pointed out in Taillard (1995). In this paper, we use the formulation (1) rather than the general QAP formulation with successful results.

3 The branch and bound algorithm

Consider the matrix of symmetric distances d_{ij} of size n by n with a zero diagonal (see Fig. 1). Suppose that m indices $p_1, p_2,..., p_m$ are selected for the cluster. Half the value of the objective function can be calculated by summing the distances in columns $p_1, p_2,...p_m$ whose rows belong to a lower p_j. These distances are all in the upper triangle of the matrix. All the combinations of selecting m columns out of n are implicitly scanned. A lower bound is constructed for every partial selection of columns. The first column is selected and the lower bound calculated, then the second column is selected and the lower bound calculated, and so on until the lower

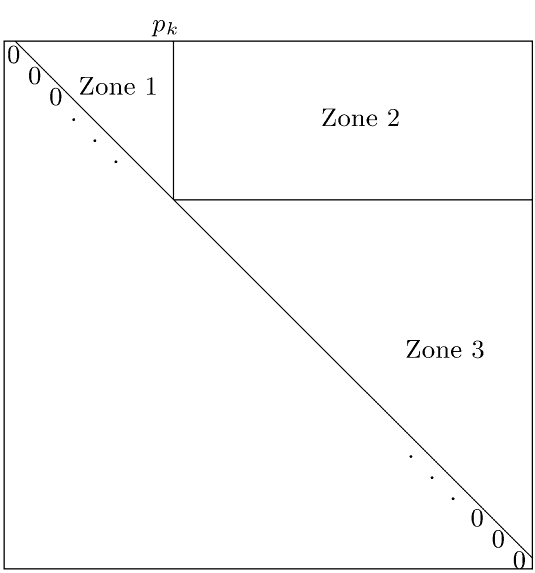

Fig. 1 Calculating the lower bounds

bound is greater than or equal to the best found solution. When the lower bound of the last selected column is greater than or equal to the best found solution, the last selected column is advanced by one place. Suppose $k \le m$ columns have been selected. If the last column is at column number $n-m+k$, no further advancement is necessary, and the next to last selected column $(k-1)$ is advanced one place, the selection of k is ignored and the process continues until the first selected column is column number $n-m+1$.

3.1 The three lower bounds

We first detail the calculation of several values which are used in the proposed lower bounds. Suppose that $k \le m$ columns $p_1 < p_2 ... < p_k$ are selected and we wish to find a lower bound for all possible selections of the remaining $m-k$ columns (all beyond column p_k). The objective function is the sum of the "relevant" distances in zones 1, 2, and 3 (see Fig. 1).

3.1.1 Preliminary values

Let the first $r-1$ distances in column r (the values "above" the zero diagonal) be sorted in increasing order yielding the vector δ_{jr} for $j=1, \ldots, r-1$. Define D_s as a lower bound on the sum of the distances in any column if it is the $s+1$th selected column (i.e. s distances are selected from this column). For example, the first selected column, p_1, includes no distances to be added to the value of the objective function because all relevant distances are below the diagonal; the second selected column, p_2, includes only one distance to be added to the objective function, the distance in row p_1; and so on. The value of D_s is:

$$D_s = \min_{s+1 \le r \le n-m+s} \left\{ \sum_{j=1}^{s} \delta_{jr} \right\} \tag{3}$$

The value of the objective function is the sum of the selected distances in zones 1,2, and 3 (see Fig. 1). We term these sums as s_1, s_2, and s_3, respectively.

For a given partial selection $p_1 < p_2 \ldots < p_k$ we have the following bounds.

3.1.2 Bounds independent of the partial selection

Zone 1: $s_1 = \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} d_{p_i p_j}$. This value is the same for all possible selections of the remaining $m-k$ columns.

Zones 2&3: $s_2 + s_3 \ge B_{23}(k) = \sum_{i=k}^{m-1} D_i$. The values of $B_{23}(k)$ for every $1 \le k \le m$ can be calculated before the branch and bound procedure and need not be repeated for any partial selection.

Zone 3: $s_3 \geq B_3(k) = \sum_{i=1}^{m-k-1} D_i$. The value of B_3 (k) can also be calculated for every k and its value is independent of the partial selection.

3.1.3 Bounds dependent on the partial selection

These bounds need to be recalculated for each partial selection.

Zone 2: For every $r > p_k$ calculate $v_r = \sum_{i=1}^{k} d_{p_i r}$. Sort the values $\{v_r\}$ in increasing order and let $\{u_r\}$ be the sorted vector. Then, $s_2 \geq B_2(k) = \sum_{i=1}^{m-k} u_i$. This bound depends on the partial selection and need to be recalculated for every partial selection.

Zones 2&3: For every $r > p_k$ sort the distances in column r starting at row p_k+1 and ending at row r (including the zero diagonal value). The sorted vector is $\{w_{ir}\}$ for i=1, ..., r−p_k. Let $D_i^{(1)} = \min_{p_k+i \leq r \leq n} \left\{ u_r + \sum_{j=1}^{i} w_{jr} \right\}$.

Then, $s_2 + s_3 \geq B_{23}^{(1)}(k) = \sum_{i=1}^{m-k} D_i^{(1)}$.

3.1.4 The suggested lower bounds

LB_1: $LB_1 = s_1 + B_{23}(k)$. This lower bound is calculated quickly because $B_{23}(k)$ need not be recalculated for every partial selection. However, this bound may not be very tight.

LB_2 : $\mathrm{LB}_2 = s_1 + \mathrm{B}_{23}^{(1)}(k)$. This lower bound is tighter than LB_1 but it takes longer to calculate because $B_{23}^{(1)}(k)$ needs to be recalculated for every partial selection. One possible weakness of LB_2 is that in the event that a column in Zone 2 has relatively low values, it may be selected repeatedly in calculating the components of $B_{23}^{(1)}(k)$.

LB_3: $LB_3 = s_1 + B_2(k) + B_3(k)$. This lower bound is also tighter than LB_1 but it takes longer to calculate because $B_2(k)$ needs to be recalculated for every partial selection. An advantage of LB_3 is that $B_2(k)$ is indeed the lowest possible value of s_2. However, the bound for s_3 is not necessarily calculated by using the same columns and may not be that tight. For larger k's the contribution of Zone 3 to the objective function is diminished. Therefore LB_3 is effective for larger k's.

Note that s_1 needs to be calculated for each lower bound. $B_{23}(k)$ is calculated once before the branch and bound procedure for every k so LB_1 is the quickest one to calculate. Therefore, when applying LB_2 or LB_3, LB_1 is calculated at no extra effort and if it is greater than or equal to the best found solution, there is no need to calculate the lower bound LB_2 or LB_3.

Table 1 Times for finding the optimal solution to Tai64c

Method	Time (min.)
Total enumeration	9,367.20
LB_1	277.54
LB_2	313.28
LB_3	131.76

Table 2 The optimal solution to $n=256$ grey pattern problems

m	Optimum	Time (min.)		
		LB_1	LB_2	LB_3
3	7,810	0.00	0.00	0.00
4	15,620	0.00	0.05	0.02
5	38,072	0.65	3.24	2.65
6	63,508	18.78	68.87	51.74
7	97,178	486.82	1,285.05	836.70
8	131,240	3,229.81	**	4,065.73

**Not attempted

3.2 Optimal solutions to grey pattern problems

We first experimented with the grey pattern Tai64c problem. In Table 1 we report the run times required to prove that the best known value of 1,855,928 is indeed optimal by total enumeration, and branch and bound using each of the bounds. The shortest run time was required by using a branch and bound procedure applying the lower bound LB_3.

We then used the branch and bound algorithm to solve grey pattern problems with $n=256$ and small values of m between 3 and 8. The run times are depicted in Table 2. All best known solutions for these problems are proven optimal. The lowest run time was required for LB_1.

4 Heuristic algorithms

We propose five heuristic solution procedures: Greedy, Steepest Descent, Tabu Search, Simulated Annealing, and an Evolutionary Algorithm. The first two algorithms take a very short run time but the three metaheuristics tend to provide higher quality solutions.

4.1 A greedy algorithm

The greedy algorithm starts with a point (the "kernel" point) and builds the cluster by adding one by one points close to the cluster. Every point is tested as the kernel point and the best cluster is selected as the result of the greedy algorithm.

The following is repeated n times, once for each point being the "kernel" point.

1. The selected cluster consists of the kernel point.
2. Go over all the points which are not in the cluster and evaluate the sum of all distances between them and all the points in the cluster.
3. Add to the cluster the point with the minimum sum of distances to all cluster points.
4. Go to Step 2 unless the cluster contains m points.
5. If the cluster contains m points, stop.

The best cluster obtained by all n possible kernel points is the result of the Greedy algorithm. The greedy algorithm requires $O(m^2n^2)$ time.

4.2 A steepest descent algorithm

The steepest descent algorithm is very similar to the algorithm proposed by Teitz and Bart (1968) for the heuristic solution of the p-median problem.

1. Select a starting cluster M of m points.
2. Evaluate the change in the value of the objective function for all possible exchanges between a point in the cluster and a point not in the cluster.
3. If there is an improving exchange, perform the best improving exchange and go to Step 2; otherwise go to Step 4.
4. Stop the algorithm with the final cluster as the solution.

4.2.1 The short cut

Note that evaluating the difference in the value of the objective function can be expedited by the following approach. Define $S_i(M) = \sum_{j=1}^{m} d_{ip_j}$. The n values of $S_i(M)$ are calculated once at the beginning of the descent algorithm for the starting solution M. This requires $O(mn)$ time and is performed once. In each iteration

1. Evaluate the change in the value of the objective function by a removal of a point $k \in M$ from the cluster and adding a point $l \notin M$ to the cluster. The change in the value of the objective function is $S_l(M) - S_k(M) - d_{kl}$. This takes O (1) time and is evaluated $m(n-m)$ times, once for each possible exchange.
2. Suppose that the selected move is to remove k from the cluster and add l to the cluster forming cluster M'. Then, $S_i(\mathrm{M}') = S_i(M) + d_{il} - d_{ik}$. The calculation of all $S_{\mathrm{i}}(M)$ requires $O(n)$ time.
3. An iteration thus requires $O(m(n-m))$ time.

The short cut is more efficient than evaluating the $m(n-m)$ exchanges directly. The calculation of an exchange directly requires $O(m)$ time for a total time of $O(m^2(n-m))$ per iteration.

4.3 A tabu search

Tabu search is a commonly used metaheuristic (Glover 1986; Glover and Laguna 1997; Salhi 1998). The parameters for the tabu search applied in this paper are: the definition of the tabu list as points recently removed from the cluster, the tabu tenure TT which is randomly generated each iteration in $[T_{\min}, T_{\max}]$, and the number of iterations, N. We also employ the stipulation that if the best found solution is improved in an iteration, the tabu list is emptied. The details of the tabu search are:

1. Select a starting solution. Its value of the objective function is the best found solution.
2. Empty the tabu list.
3. If the number of iterations exceeds N, stop with the best found solution as the result of the algorithm.
4. Randomly generate the tabu tenure TT in $[T_{\min}, T_{\max}]$.
5. Evaluate all $m(n-m)$ possible exchanges.
6. If an exchange leads to a solution better than the best found solution, perform the best exchange, update the best found solution, and go to Step 2; otherwise go to Step 7.
7. – Perform the best exchange (whether improving or not) excluding exchanges for which the tenure of the entering point in the tabu list is less than the tabu tenure TT.
   – Enter the point removed from the cluster to the tabu list.
   – Go to Step 3.

Since all $m(n-m)$ possible exchanges are evaluated every iteration in Step 5, the short cut used for the steepest descent algorithm (Section 4.2.1) is used also for the tabu search.

4.4 A simulated annealing algorithm

Simulated annealing is also a commonly used metaheuristic. It was suggested by Kirkpatrick et al. (1983) and simulates the annealing of metals from a high temperature liquid to a low temperature solid (see also Glover and Laguna 1997; Salhi 1998). The parameters for the simulated annealing are: the starting temperature T_0, the cooling factor α, and the number of iterations N. The variant used in this paper is:

1. Set the temperature $T = T_0$. Generate a starting cluster.
2. Randomly select a point in the cluster to be removed and a point not in the cluster to be added to the cluster.
3. Calculate the change in the value of the objective function ΔF.
4. If $\Delta F \leq 0$ perform the exchange and go to Step 6.
5. Calculate $\delta = \frac{\Delta F}{T}$. Perform the exchange with probability $e^{-\delta}$. Otherwise, retain the current cluster.

6. Multiply the temperature T by α, and if the number of iterations does not exceed N, go to Step 2.
7. Stop the procedure with the best found cluster during the process as the result of the algorithm.

The simulated annealing procedure requires $O(mN)$ time. However, the calculation of $e^{-\delta}$ and the random number associated with the calculation of the probability requires quite a large proportion of the computer time. Since for $\delta > 10$ the probability of accepting a move is negligible (4.54×10^{-5}), it is assumed "0" and the probability is not calculated.

4.5 Evolutionary algorithms

A population of P member solutions is maintained. Each population member M is represented by a set of m points.

1. Randomly generate a population of P solutions.
2. Repeat the following G times (generations)

   – Randomly select two members of the population and merge them to produce an offspring M'.
   – If $F(M')$ is not better than the worst population member, do not change the population and start the next generation.
   – If $F(M')$ is better than the worst population member, then

     – If the offspring is identical to an existing population member, do not change the population and start the next generation.
     – Otherwise, replace the worst population member with M' and start the next generation.

3. The best population member is the resulting solution.

The most important part of an evolutionary algorithm is the merging process applied to produce an offspring. We tested two merging processes with a parameter K: the descent merging process and the tabu merging process.

4.5.1 The descent merging process

The descent merging process is similar to the merging process suggested in Berman and Drezner (2005).

1. The two parents are M_1 and M_2, each represented by a set of m points.
2. The intersection between M_1 and M_2 is: $M^I = M_1 \cap M_2$. The cardinality of the intersection is m^I.
3. The union of M_1 and M_2 is: $M^U = M_1 \cup M_2$. The cardinality of M^U is $2m - m^I$.
4. K different points not in M^U are selected to form M^K (if $2m - m^I + K > n$ then select only $n - 2m + m^I$ points).

5. All the points in M^U which are not in M^I define M^E. The cardinality of M^E is $2m-2m^I$.
6. Define $M^D=M^E \cup M^K$. The cardinality of M^D is $m^D=\min\{n-m^I,\ 2m-2m^I+K\}$.
7. A starting offspring M' is created by randomly adding $m-m^I$ points from M^D to M^I.
8. A restricted descent process is performed on M' by adding or removing only points in M^D and keeping the points in M^I fixed.
9. The result of the restricted descent process is the offspring.

Table 3 Best known results for the grey pattern problems

m	BKV	m	BKV	m	BKV	m	BKV
3	7,810*	35	4,890,132	67	21,439,396	99	52,660,116
4	15,620*	36	5,222,296	68	22,234,020	100	53,838,088
5	38,072*	37	5,565,236	69	23,049,732	101	55,014,262
6	63,508*	38	5,909,202	70	23,852,796	102	56,202,826
7	97,178*	39	6,262,248	71	24,693,608	103	57,417,112
8	131,240*	40	6,613,472	72	25,529,984	104	58,625,240
9	183,744	41	7,002,794	73	26,375,828#	105	59,854,744
10	242,266	42	7,390,586	74	27,235,240	106	61,084,902
11	304,722	43	7,794,422	75	28,114,952	107	62,324,634
12	368,952	44	8,217,264	76	29,000,908	108	63,582,416
13	457,504	45	8,674,910	77	29,894,452	109	64,851,966
14	547,522	46	9,129,192	78	30,797,954	110	66,120,434
15	644,036	47	9,575,736	79	31,702,182	111	67,392,724
16	742,480	48	10,016,256	80	32,593,088	112	68,666,416
17	878,888	49	10,518,838	81	33,544,628	113	69,984,758
18	1,012,990	50	11,017,342	82	34,492,592	114	71,304,194
19	1,157,992	51	11,516,840	83	35,443,938#	115	72,630,764
20	1,305,744	52	12,018,388	84	36,395,172#	116	73,962,220
21	1,466,210	53	12,558,226	85	37,378,800#	117	75,307,424
22	1,637,794	54	13,096,646	86	38,376,438	118	76,657,014
23	1,820,052	55	13,661,614	87	39,389,054	119	78,015,914
24	2,010,846	56	14,229,492	88	40,416,536	120	79,375,832
25	2,215,714	57	14,793,682	89	41,512,742	121	80,756,852
26	2,426,298	58	15,363,628	90	42,597,626#	122	82,138,768
27	2,645,436	59	15,981,086	91	43,676,474#	123	83,528,554
28	2,871,704	60	16,575,644	92	44,759,294	124	84,920,540
29	3,122,510	61	17,194,812	93	45,870,244#	125	86,327,812
30	3,373,854	62	17,822,806	94	46,975,856#	126	87,736,646
31	3,646,344	63	18,435,790	95	48,081,112	127	89,150,166
32	3,899,744	64	19,050,432	96	49,182,368	128	90,565,248
33	4,230,950	65	19,848,790	97	50,344,050		
34	4,560,162	66	20,648,754	98	51,486,642		

*Optimal
#A new BKV

Note that the shortcut described in Section 4.2.1 can be applied to the restricted descent process.

4.5.2 The tabu merging process

We also experimented with a tabu extension of the restricted descent search. Let h be the number of iterations performed by the restricted descent algorithm. A restricted tabu search of $5h$ iterations is performed following the approach outlined in Section 4.3. We need to select $m-m^I$ points out of m^D points. If $m^D-(\mathrm{m}-m^I)\leq 5$,

Table 4 Results for the grey pattern problems by tabu search

m	2,000n		10,000n		m	2,000n		10,000n		m	2,000n		10,000n	
	†	‡	†	‡		†	‡	†	‡		†	‡	†	‡
22	100	0	100	0	53	71	0.009	100	0	84	0	0.060	0	0.040
23	100	0	100	0	54	87	0.001	100	0	85	0	0.089	0	0.067
24	100	0	100	0	55	43	0.010	80	0.004	86	0	0.081	0	0.046
25	100	0	100	0	56	92	0.002	98	0.001	87	0	0.128	0	0.069
26	29	0.009	75	0.003	57	9	0.037	9	0.034	88	0	0.197	5	0.150
27	100	0	99	0.002	58	58	0.037	76	0.022	89	0	0.114	1	0.081
28	100	0	100	0	59	8	0.032	18	0.030	90	0	0.076	0	0.061
29	100	0	100	0	60	99	0.000	100	0	91	0	0.069	0	0.057
30	100	0	100	0	61	100	0	100	0	92	2	0.058	9	0.038
31	100	0	100	0	62	100	0	100	0	93	4	0.024	9	0.009
32	100	0	100	0	63	100	0	100	0	94	1	0.020	2	0.011
33	100	0	100	0	64	100	0	100	0	95	47	0.002	97	0.000
34	70	0.014	70	0.011	65	0	0.114	0	0.121	96	100	0	100	0
35	100	0	100	0	66	0	0.079	0	0.088	97	100	0	100	0
36	94	0.001	95	0.001	67	12	0.041	8	0.041	98	100	0	100	0
37	100	0	100	0	68	1	0.037	10	0.024	99	100	0	100	0
38	100	0	100	0	69	1	0.047	12	0.027	100	100	0	100	0
39	100	0	100	0	70	3	0.126	11	0.085	101	100	0	100	0
40	100	0	100	0	71	1	0.090	2	0.059	102	97	0.001	100	0
41	98	0.006	97	0.008	72	0	0.090	1	0.052	103	93	0.000	100	0
42	100	0	100	0	73	0	0.092	0	0.073	104	14	0.004	44	0.002
43	64	0.013	62	0.010	74	0	0.087	0	0.066	105	100	0	100	0
44	14	0.035	55	0.013	75	0	0.058	1	0.037	106	99	0.000	100	0
45	5	0.045	25	0.023	76	0	0.048	1	0.034	107	100	0	100	0
46	2	0.036	18	0.017	77	0	0.049	1	0.033	108	100	0	100	0
47	46	0.016	97	0.001	78	0	0.034	0	0.024	109	100	0	100	0
48	67	0.074	100	0	79	10	0.034	18	0.022	110	99	0.000	100	0
49	36	0.057	91	0.004	80	56	0.028	75	0.017	111	100	0	100	0
50	43	0.004	94	0.000	81	87	0.000	100	0	112	100	0	100	0
51	15	0.003	45	0.000	82	1	0.015	3	0.012					
52	37	0.023	82	0.001	83	0	0.018	0	0.008	Ave	55.1	0.027	62.6	0.018

†Number of times out of 100 that BKV found
‡Percentage of average solution over BKV

the tabu search is not performed and the result of the descent algorithm is applied. Note that the shortcut described in Section 4.2.1 can be applied to the restricted tabu search.

These merging processes do not resemble neither the standard crossover operator nor the hybrid genetic algorithm approach. However, they combine elements of both and thus the procedure is termed "an evolutionary algorithm" because it does not conform to the standard genetic or hybrid genetic algorithms.

5 Computational experiments with heuristic algorithms

We first solved the 126 grey pattern problems with $n=256$ (Taillard and Gambardella 1997). Optimal solutions were obtained for problems with $3 \leq m \leq 8$ (see Table 2), and the rest were solved by heuristic algorithms. We then used the 40 problems suggested by Beasley (1990) for the p-median problem on a network, and solved them as cluster problems.

5.1 Grey pattern problems

We present the results in non-chronological order because values presented in later tables require the value of the best known value (BKV) depicted in Table 3. All BKVs where obtained by the evolutionary algorithm with the tabu merging process

Table 5 Summary of experiments with ten replications per problem

K=	0	1	2	3	5	7	10	15	20
Descent merging process (all 91 problems)									
(1)	0.118	0.083	0.078	0.078	0.069	0.065	0.067	0.064	0.056
(2)	30	41	53	48	51	54	54	54	55
(3)	61	138	180	189	213	230	234	243	279
(4)	1	3	2	1	3	2	2	3	3
(5)	260.1	259.4	267.2	292.0	333.5	358.6	418.2	521.4	618.9
Tabu merging process (all 91 problems)									
(1)	0.039	0.019	0.016	0.013	0.012				
(2)	78	83	85	90	87				
(3)	296	525	550	575	598				
(4)	6	8	7	8	8				
(5)	2,333.2	2,440.5	2,746.2	2,870.5	3,550.3				
Tabu merging process ($72 \leq m \leq 94$)									
(1)	0.026	0.017	0.017	0.015	0.016				
(2)	17	19	18	22	20				
(3)	59	92	96	107	100				
(4)	6	8	7	8	8				
(5)	841.9	816.3	887.4	934.1	1,061.1				

(1) Percent of average over BKV (%); (2) Number of problems for which BKV found; (3) Number of times BKV found; (4) Problems for which a new BKV found; (5) Total time for all runs (min)

and adding K=3 dots. The original best known values are reported in Taillard and Gambardella (1997). Misevicius (2003a,b, 2004, 2005) improved some best known values. Eight new improved BKVs are reported in Table 3.

The greedy heuristic solved all 126 problems in less than 2 min. It found the BKV for 11 problems (m=3, 4, 8, 16, 112, 120, 124–128). The average was 1.437% over the BKV.

The descent approach was replicated 1,000 times for each problem. Solving all 126 problems took about 4 min. It found the BKV at least once in 1,000 trials for 25 of the problems (m=3–17, 20, 21, 23, 24, 31, 33, 123, 125, 126, 128). The minimum value in 1,000 trials was, on the average, 0.250% above the BKV.

Table 6 Results for the grey pattern problems by the evolutionary algorithm

m	Descent		Tabu		m	Descent		Tabu		m	Descent		Tabu	
	†	‡	†	‡		†	‡	†	‡		†	‡	†	‡
22	99	0.000	100	0	53	29	0.055	97	0.001	84	14	0.011	76	0.001
23	100	0	100	0	54	2	0.112	87	0.004	85	11	0.013	39	0.001
24	99	0.000	99	0.000	55	0	0.085	48	0.010	86	14	0.015	80	0.000
25	87	0.004	95	0.001	56	4	0.056	70	0.005	87	28	0.035	94	0.000
26	7	0.026	8	0.021	57	13	0.078	98	0.000	88	3	0.109	78	0.014
27	69	0.021	90	0.006	58	5	0.195	91	0.005	89	12	0.056	62	0.006
28	46	0.046	49	0.040	59	7	0.064	81	0.005	90	2	0.066	60	0.006
29	81	0.007	96	0.001	60	11	0.117	33	0.039	91	0	0.083	66	0.007
30	89	0.023	100	0	61	7	0.085	55	0.021	92	0	0.099	65	0.010
31	44	0.170	78	0.055	62	16	0.022	83	0.003	93	0	0.079	36	0.005
32	61	0.252	68	0.124	63	22	0.076	91	0.003	94	0	0.091	16	0.016
33	69	0.023	94	0.004	64	20	0.206	72	0.028	95	0	0.122	40	0.026
34	36	0.030	49	0.019	65	16	0.105	66	0.019	96	0	0.185	29	0.065
35	44	0.020	74	0.006	66	21	0.033	47	0.013	97	0	0.125	30	0.040
36	29	0.019	61	0.005	67	4	0.070	29	0.028	98	0	0.129	13	0.052
37	77	0.004	99	0.000	68	0	0.126	18	0.059	99	0	0.104	32	0.020
38	93	0.003	100	0	69	0	0.108	42	0.025	100	0	0.082	42	0.005
39	94	0.001	100	0	70	0	0.182	28	0.079	101	0	0.092	46	0.012
40	81	0.016	96	0.001	71	1	0.132	19	0.034	102	0	0.095	38	0.012
41	74	0.005	97	0.001	72	1	0.124	1	0.039	103	0	0.064	60	0.002
42	63	0.007	95	0.001	73	0	0.128	17	0.057	104	0	0.065	9	0.008
43	51	0.009	87	0.001	74	0	0.112	3	0.062	105	1	0.064	67	0.001
44	8	0.159	24	0.107	75	1	0.075	25	0.020	106	1	0.080	34	0.002
45	8	0.034	24	0.018	76	0	0.064	13	0.010	107	7	0.085	90	0.001
46	20	0.019	42	0.005	77	0	0.053	13	0.016	108	10	0.070	82	0.000
47	57	0.038	92	0.004	78	0	0.041	6	0.013	109	28	0.046	92	0.000
48	49	0.150	90	0.021	79	2	0.050	13	0.022	110	19	0.042	83	0.000
49	32	0.083	83	0.018	80	3	0.107	27	0.058	111	27	0.035	87	0.000
50	59	0.007	100	0	81	13	0.032	65	0.004	112	22	0.043	79	0.001
51	26	0.004	80	0.000	82	16	0.011	52	0.002					
52	27	0.052	84	0.002	83	26	0.006	90	0.000	Ave	24.4	0.067	61.1	0.016

†Number of times out of 100 that BKV found
‡Percentage of average solution over BKV

We tried many possible parameters for the simulated annealing approach but could not find good parameters for it. We therefore report only results obtained by the tabu search and the evolutionary algorithm. Since problems with $m \leq 21$ and $m \geq 113$ are easily solved by all metaheuristic approaches, we report in the rest of this section results of experiments with problems of $22 \leq m \leq 112$ dots.

5.1.1 Experiments with tabu search

We tried various values for $T_{\min}$ and $T_{\max}$ and following the experience in Drezner and Marcoulides (2005) and our own extensive experiments we applied

Table 7 Run times (min/run) for the grey pattern problems

m	(1)	(2)	(3)	(4)	*m*	(1)	(2)	(3)	(4)	*m*	(1)	(2)	(3)	(4)
22	0.21	0.96	0.23	1.12	53	0.41	1.90	0.38	3.21	84	0.56	2.54	0.43	2.96
23	0.22	1.00	0.20	1.11	54	0.42	1.92	0.40	2.91	85	0.56	2.55	0.42	3.43
24	0.22	1.03	0.19	0.94	55	0.42	1.95	0.47	4.92	86	0.57	2.57	0.41	2.70
25	0.23	1.06	0.30	1.62	56	0.42	1.97	0.42	3.90	87	0.57	2.58	0.35	2.11
26	0.24	1.10	0.32	1.67	57	0.43	1.99	0.36	2.79	88	0.58	2.60	0.35	2.20
27	0.25	1.13	0.31	1.66	58	0.43	2.02	0.40	3.60	89	0.58	2.61	0.38	2.89
28	0.25	1.16	0.33	1.78	59	0.44	2.04	0.37	3.91	90	0.58	2.63	0.42	5.04
29	0.26	1.20	0.30	1.57	60	0.44	2.07	0.40	3.97	91	0.59	2.64	0.48	5.35
30	0.27	1.23	0.28	1.54	61	0.45	2.09	0.46	6.43	92	0.59	2.66	0.48	3.73
31	0.27	1.26	0.22	1.40	62	0.45	2.11	0.35	1.66	93	0.59	2.67	0.47	4.34
32	0.28	1.29	0.21	1.26	63	0.46	2.13	0.39	6.35	94	0.59	2.69	0.48	4.55
33	0.29	1.32	0.13	0.98	64	0.47	2.16	0.49	8.60	95	0.59	2.70	0.48	3.61
34	0.29	1.36	0.18	1.01	65	0.47	2.19	0.47	6.93	96	0.59	2.71	0.52	3.94
35	0.30	1.39	0.30	1.73	66	0.47	2.21	0.32	3.95	97	0.59	2.73	0.46	2.59
36	0.30	1.42	0.31	1.75	67	0.47	2.23	0.32	5.09	98	0.60	2.73	0.45	2.42
37	0.31	1.45	0.28	1.69	68	0.47	2.25	0.36	4.07	99	0.60	2.75	0.49	2.81
38	0.31	1.48	0.27	1.51	69	0.48	2.27	0.43	4.74	100	0.60	2.76	0.44	1.81
39	0.32	1.51	0.21	1.12	70	0.48	2.29	0.44	4.26	101	0.60	2.77	0.42	2.08
40	0.33	1.54	0.17	1.80	71	0.5	2.31	0.53	4.77	102	0.61	2.78	0.40	1.60
41	0.34	1.57	0.26	2.28	72	0.5	2.33	0.58	4.95	103	0.61	2.79	0.38	1.36
42	0.34	1.60	0.30	2.39	73	0.51	2.34	0.62	5.59	104	0.61	2.81	0.43	1.91
43	0.35	1.62	0.49	3.32	74	0.51	2.36	0.69	5.97	105	0.61	2.83	0.47	1.71
44	0.36	1.65	0.46	3.35	75	0.52	2.38	0.69	5.72	106	0.62	2.84	0.56	1.88
45	0.36	1.68	0.44	3.44	76	0.52	2.40	0.68	5.31	107	0.63	2.86	0.52	2.04
46	0.37	1.71	0.37	2.70	77	0.52	2.42	0.70	6.26	108	0.64	2.87	0.39	2.98
47	0.37	1.74	0.23	1.30	78	0.53	2.44	0.65	5.03	109	0.64	2.89	0.36	1.45
48	0.38	1.77	0.25	1.76	79	0.54	2.45	0.58	4.22	110	0.64	2.91	0.33	3.91
49	0.38	1.79	0.37	3.02	80	0.54	2.47	0.48	2.96	111	0.64	2.92	0.29	5.96
50	0.39	1.82	0.36	3.22	81	0.54	2.49	0.44	1.56	112	0.64	2.93	0.30	13.16
51	0.39	1.84	0.36	3.45	82	0.55	2.51	0.42	2.07					
52	0.40	1.87	0.37	3.23	83	0.55	2.52	0.46	2.78	Ave	0.46	2.12	0.40	3.25

(1) Tabu with 2000n iterations; (2) Tabu with 10000n iterations; (3) Evolutionary with descent merging; (4) Evolutionary with tabu merging

successfully a wide range for the tabu tenure with $T_{\min}=0.2(n-m)$ and $T_{\max}=0.2(n-m)$. We tested the tabu search with $N=2{,}000n$ and $N=10{,}000n$ iterations. Each problem was solved 100 times. The computational results are depicted in Table 4 (run times are given in Table 7). There seem to be three distinct "regions" of m. Problems with $m\leq 64$ and problems with $m\geq 95$ are, with a few exceptions, easily solved by the tabu search. Problems with $65\leq m\leq 94$, with a few exceptions, seem to be difficult for tabu search. Overall, tabu search with $N=10{,}000n$ iterations failed to find the BKV in 12 problems but found the BKV in all 100 replications for 41 problems (see Table 8). Increasing the number of iterations from $2{,}000n$ to $10{,}000n$ improved the performance of the algorithm. However, we believe that diversification approaches may further improve the performance of the algorithm without adding run time.

5.1.2 Experiments with the evolutionary algorithm

We tested the evolutionary algorithm with both suggested merging processes. Following extensive experiments we set the population size at 300 (slightly higher than $n=256$) and the number of generations to $1{,}000n=256{,}000$. In order to determine the preferred value of the parameter K, we solved the 91 problems ($22\leq m\leq 112$) ten times for various values of K. The results are summarized in Table 5. For the descent merging process, the performance seems to improve with the increase in the value of K. However, run time increases as well. By inspection of the table we selected $K=7$ for further experiments. We also depict in Table 5

Table 8 Summary of grey pattern problems ($22\leq m\leq 112$)

	Tabu		Evolutionary	
	$2{,}000n$	$10{,}000n$	Descent	Tabu
Frequency of BKV				
None	18	12	23	0
0–10	31	26	43	5
11–20	4	5	11	8
21–30	1	1	2	8
31–40	2	0	4	7
41–50	4	2	3	8
51–60	2	1	4	4
61–70	3	2	2	9
71–80	1	4	4	8
81-90	2	1	4	13
91–100	41	49	5	21
100	33	41	1	6
New	2	2	4	8
Averages				
# BKV	55.1	62.6	24.4	61.1
Percent over BKV (%)	0.027	0.018	0.067	0.016
Time (min)	0.46	2.12	0.40	3.25

Table 9 Cluster solutions for branch and bound, greedy, and descent

n	m	Best known	B&B		Greedy		Descent		
			†	Time‡	†	Time‡	#	$	Time‡
100	5	260*	0	0.01	0	0.01	410	71.065	0.02
100	10	2,664*	0	0.19	0	0.00	522	4.879	0.04
100	10	1,856*	0	0.00	0	0.00	477	76.154	0.04
100	20	29,322*	0	2,000.59	0	0.00	462	4.995	0.11
100	33	74,032	0	3,600.01	0.159	0.02	390	0.132	0.21
200	5	92*	0	0.01	0	0.00	131	112.913	0.04
200	10	862*	0	0.44	0	0.01	732	19.432	0.10
200	20	8,790*	0	163.25	0	0.03	954	2.630	0.27
200	40	64,500	0.465	3,600.00	0.598	0.10	236	1.367	0.75
200	67	161,884	0	3,600.00	0.033	0.24	381	0.029	1.41
300	5	68*	0	0.01	0	0.01	18	63.103	0.06
300	10	858*	0	0.81	0	0.02	225	39.606	0.14
300	30	17,052	0	3,600.00	0	0.13	251	1.035	0.87
300	60	116,480	0.539	3,600.00	0.539	0.46	114	0.214	2.49
300	100	323,024	0	3,600.00	0.016	2.22	574	0.002	4.74
400	5	44*	0	0.03	0	0.01	140	92.636	0.14
400	10	496*	0	1.25	0	0.03	398	41.272	0.32
400	40	33,104	0.278	3,600.00	0.296	0.38	230	0.690	2.45
400	80	155,784	0.134	3,600.00	0.153	1.97	108	0.015	6.72
400	133	533,392	0	3,600.00	0.001	8.93	538	0.001	12.48
500	5	40*	0	0.03	0	0.02	34	109.475	0.28
500	10	608*	0	18.94	0	0.05	100	19.056	0.70
500	50	42,526	0	3,600.00	0	0.92	931	0.128	5.57
500	100	226,950	0.057	3,600.00	0.061	6.79	99	0.019	15.43
500	167	638,878	0.014	3,600.02	0.046	25.78	1,000	0	32.63
600	5	36*	0	0.06	0	0.02	107	99.694	0.42
600	10	312*	0	2.86	0	0.07	339	21.369	0.94
600	60	49,250	0.374	3,600.00	0.459	1.90	1,000	0	10.28
600	120	268,154	0.481	3,600.09	0.481	17.18	258	0.009	30.82
600	200	937,014	0	3,600.06	0	59.72	990	0.001	69.71
700	5	32*	0	0.08	0	0.03	31	86.938	0.56
700	10	330*	0	12.00	0	0.10	270	44.706	1.32
700	70	74,524	0	3,600.02	0	3.90	558	0.313	16.88
700	140	336,418	0.015	3,600.00	0.026	36.45	597	0.008	54.09
800	5	36*	0	0.14	0	0.03	199	49.806	0.67
800	10	334*	0	32.47	0	0.12	122	29.892	1.48
800	80	96,124	0.501	3,600.03	0.504	8.66	631	0.122	26.08
900	5	32*	0	0.17	0	0.05	52	48.375	0.78
900	10	264*	0	24.58	0	0.14	353	25.769	1.76
900	90	112,444	0.004	3,600.05	0.004	18.00	244	0.445	40.78
Average			0.072	1,766.46	0.084	4.86	380.2	26.707	8.61

†Percent of value over best known solution
‡Time in seconds for all runs
#Number of times best known solution found out of 1,000 replications
$Percent of average over the best known solution
*Optimal

experiments with various values of K for the tabu merging process. There seem to be a drop in performance when K is increased from $K=3$ to $K=5$. To confirm this phenomenon, we also report the performance of the algorithm for problems with $72 \le m \le 94$. This range includes all new BKVs, and seems to contain more difficult problems for this approach. Consequently, we selected $K=3$ for further experiments with the tabu merging process. The results of 100 replications for each problem are reported in Table 6 (and run times reported in Table 7). The results are much better for the tabu merging process but run times are about eight times longer for the tabu merging process. The descent merging algorithm missed the BKV for 23 problems while the tabu merging algorithm found it at least once for all problems. The average number of found BKV increased from about 24 to 61% (see Table 8). The average solution was 0.067% over the BKV for the descent merging algorithm and only 0.016% for the tabu merging algorithm.

5.2 Cluster problems

We tested the cluster problems on the 40 problems given in Beasley (1990) for the p-median problem. These problems range between $n=100$ and $n=900$ points, and the cluster sizes range between $m=5$ and $m=200$ points. We were unsuccessful in finding good parameters for the tabu search so we report in Table 9 only results for the branch and bound with LB_3 (which was the best variant), the greedy algorithm, and the steepest descent approach. In Table 10, we report the results for simulated annealing (with the following parameters: $T_0=1{,}000$, $N=50{,}000n$, and $\alpha = 1 - \frac{10}{N}$), and the evolutionary algorithm using the tabu merging process with a population size of $P=n$ and $1{,}000n$ generations. The branch and bound procedure was terminated after 1 h if it did not finish, the descent algorithm was replicated 1,000 times, and the simulated annealing and the evolutionary algorithm were replicated ten times each.

Examining Tables 9 and 10, we conclude that 21 of the 40 problems were solved to optimality within 1 h of computer time. Twenty nine problems terminated with the best known solution. Overall, the final solution of the branch and bound algorithm was 0.072% above the best known solution. The greedy algorithm found the best known solution for 25 of the 40 problems. The solution of the greedy algorithm was, on the average, 0.084% over the best known solution. The descent approach and the simulated annealing found the best known solution at least once for all 40 problems. The evolutionary algorithm failed to find the BKV for one problem. The run time for 1,000 replications of the descent approach was faster than the run time for ten replications of the simulated annealing or the evolutionary algorithm by a factor of about 40. This means that for "fair" comparison we have to run the descent algorithm 40,000 times. The descent algorithm found the best known solution in about 38% of the replications while simulated annealing found it in almost 99% of the cases. The evolutionary algorithm found it in about 94% of the cases. We conjecture that all best known solutions are optimal. Twenty one problems were solved to optimality. The descent algorithm found the best known solution at least 99 times out of 1,000 replications and the simulated annealing algorithm found it at least nine times out of 10 for the remaining 19 problems. The recommended procedure is the descent algorithm which seems to perform very

Table 10 Cluster solutions for simulated annealing and evolutionary algorithm

n	m	Best known	Simulated annealing			Evolutionary		
			#	$	Time‡	#	$	Time‡
100	5	260*	10	0	15.65	10	0	9.47
100	10	2,664*	10	0	16.66	10	0	13.42
100	10	1,856*	10	0	16.57	10	0	6.98
100	20	29,322*	10	0	18.42	6	1.101	39.76
100	33	74,032	10	0	21.80	10	0	12.19
200	5	92*	10	0	32.65	10	0	34.39
200	10	862*	10	0	34.90	10	0	41.53
200	20	8,790*	10	0	38.51	10	0	23.17
200	40	64,500	10	0	47.96	7	0.034	166.08
200	67	161,884	10	0	53.12	10	0	52.76
300	5	68*	10	0	51.90	9	0.588	47.15
300	10	858*	10	0	55.46	10	0	42.70
300	30	17,052	9	0.137	76.59	9	0.110	1,103.82
300	60	116,480	10	0	88.09	10	0	82.44
300	100	323,024	10	0	99.25	10	0	126.90
400	5	44*	10	0	84.12	10	0	101.43
400	10	496*	10	0	93.12	10	0	62.07
400	40	33,104	10	0	150.89	10	0	106.13
400	80	155,784	10	0	193.26	10	0	185.10
400	133	533,392	10	0	229.26	10	0	299.30
500	5	40*	9	1.000	124.11	10	0	124.36
500	10	608*	9	0.428	148.98	10	0	247.09
500	50	42,526	10	0	279.17	10	0	158.78
500	100	226,950	10	0	380.57	9	0.000	360.14
500	167	638,878	10	0	466.45	10	0	602.53
600	5	36*	10	0	166.92	9	1.111	174.46
600	10	312*	10	0	207.87	10	0	139.06
600	60	49,250	10	0	446.10	10	0	234.53
600	120	268,154	10	0	636.95	10	0	589.21
600	200	937,014	10	0	760.18	10	0	1,271.97
700	5	32*	10	0	210.15	10	0	246.53
700	10	330*	10	0	278.10	10	0	173.98
700	70	74,524	10	0	730.31	10	0	3,604.65
700	140	336,418	10	0	940.74	10	0	1,041.27
800	5	36*	10	0	252.63	10	0	355.36
800	10	334*	8	0.838	332.53	10	0	454.70
800	80	96,124	10	0	979.83	10	0	554.58
900	5	32*	10	0	300.78	8	2.500	382.44
900	10	264*	10	0	414.25	10	0	255.64
900	90	112,444	10	0	1,366.54	0	0.366	1,419.07
Average			9.88	0.060	271.03	9.43	0.145	373.68

†Percent of value over best known solution
‡Time in seconds for all runs
#Number of times best known solution found out of 1000 replications
$Percent of average over the best known solution
*Optimal

well in an extremely short computer time (less than half a second required for the largest problem). Simulated annealing requires much longer time (about 2 min for the largest problem) but finds the BKV in a large proportion of the cases. The Evolutionary algorithm, while performing quite well, is inferior to the simulated annealing algorithm both in the quality of the solution and run time.

6 Conclusions

We presented the cluster selection problem and proposed three robust algorithms and five heuristic algorithms for its solution. The same algorithms are also applied for the solution of the grey pattern problem which is usually formulated as a quadratic assignment problem. The computational results demonstrated the efficiency and effectiveness of the proposed solution methods.

As future research, we suggest to examine possible improvements to the lower bounds. We experimented with bounds that depend on the values of k and p_k (the last column in Zone 1) which can be calculated once before the branch and bound process. Such bounds for Zones 2 and 2&3 may improve the performance of the branch and bound. However, in our experiments we did not observe significantly improved performance. It is also suggested to further investigate metaheuristic algorithms for the solution of these problems and employ a diversification strategy for the tabu search.

References

Beasley JE (1990) OR-library—distributing test problems by electronic mail. J Oper Res Soc 41:1069–1072. Also available at http://mscmga.ms.ic.ac.uk/jeb/orlib/pmedinfo.html

Berman O, Drezner Z (2005) The multiple server location problem. J Oper Res Soc (in press)

Current J, Daskin M, Schilling D (2002) Discrete network location models. Ch. 3 In: Drezner Z, Hamacher HW (eds) Location analysis: applications and theory, pp 81–118

Daskin M (1995) Network and discrete location: models, algorithms and applications. Wiley, New York

Drezner Z (1981) On a modified one-center model. Manage Sci 27:848–851

Drezner Z, Hahn PM, Taillard ED (2005) Recent advances for the quadratic assignment problem with special emphasis on instances that are difficult for meta-heuristic methods. Ann Oper Res 139:65–94

Drezner Z, Marcoulides GA (2005) On the range of tabu tenure in solving quadratic assignment problems. Under review

Glover F (1986) Future paths for integer programming and links to artificial intelligence. Comput Oper Res 13:533–549

Glover F, Laguna M (1997) Tabu search. Kluwer, Boston

Kirkpatrick S, Gelat CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–680

Koopmans TC, Beckmann MJ (1957) Assignment problems and the location of economics activities. Econometrica 25:53–76

Misevicius A (2003a) Ruin and recreate principle based approach for the quadratic assignment problem. Lecture notes in computer science, vol 2723. In: Cant-Paz E, Foster JA, Deb K, et al (eds) Genetic and evolutionary computation—GECCO 2003 (Chicago, USA), Proceedings, Part I, pp 598–609. Springer, Berlin Heidelberg New York

Misevicius A (2003b) Genetic algorithm hybridized with ruin and recreate procedure: application to the quadratic assignment problem. Knowl Based Syst 16:261–268

Misevicius A (2004) An improved hybrid genetic algorithm: new results for the quadratic assignment problem. Knowl Based Syst 17:65–73

Misevicius A (2005) A tabu search algorithm for the quadratic assignment problem. Working paper, Kaunas University of Technology, Kaunas, Lithuania Comput Optim Appl 30:95–111
Rendl F (2002) The quadratic assignment problem. In: Drezner Z, Hamacher H (eds) Facility location: applications and theory. Springer, Berlin Heidelberg New York
Salhi S (1998) Heuristic search methods. In: Marcoulides G (ed) Modern methods for business research. Lawrence Erlbaum Associates, Mahwah, NJ
Taillard ED (1995) Comparison of iterative searches for the quadratic assignment problem. Location Science 3:87–105
Taillard ED, Gambardella LM (1997) Adaptive memories for the quadratic assignment problem. Tech. report IDSIA-8797. Lugano, Switzerland
Teitz MB, Bart P (1968) Heuristic methods for estimating the generalized vertex median of a weighted graph. Oper Res 16:955-961
Watson-Gandy CDT (1982) Heuristic procedures for the M-partial cover problem on a plane. Eur J Oper Res 11:149-157