Global Equilibrium Search Algorithms for Combinatorial Optimization Problems

Oleg Shylo[1], Dmytro Korenkevych[2], and Panos M. Pardalos[2,3,⋆]

[1] Department of Industrial Engineering, University of Pittsburgh, Pittsburgh, PA 15261
[2] Department of Industrial and Systems Engineering, University of Florida, 303 Weil Hall, Gainesville, FL, 32611, USA
[3] National Research University Higher School of Economics, Laboratory of Algorithms and Technologies for Network Analysis (LATNA), 136 Rodionova St., Nizhny Novgorod 603093, Russia

Abstract. Global Equilibrium Search (GES) is a meta-heuristic framework that shares similar ideas with the simulated annealing method. GES accumulates a compact set of information about the search space to generate promising initial solutions for the techniques that require a starting solution, such as the simple local search method. GES has been successful for many classic discrete optimization problems: the unconstrained quadratic programming problem, the maximum satisfiability problem, the max-cut problem, the multidimensional knapsack problem and the job-shop scheduling problem. GES provides state-of-the-art performance on all of these domains when compared to the current best known algorithms from the literature. GES algorithm can be naturally extended for parallel computing as it performs search simultaneously in distinct areas of the solution space. In this talk, we provide an overview of Global Equilibrium Search and discuss some successful applications.

Keywords: discrete optimization, meta-heuristics, global equilibrium search.

1 Method Description

Simulated annealing (SA) [1] is a randomized metaheuristic approach that was successfully applied to a variety of discrete optimization problems. Usually, the search process under SA consists of a sequence of transitions between feasible solutions that is guided by certain probabilistic rules. Standard simulated annealing is a memoryless optimization approach – the transitions between solutions are independent from the previous search states. Global equilibrium search (GES) [2] shares similar ideas, but, unlike simulated annealing, GES uses adaptive memory structures to collect information about visited solutions, and actively uses this knowledge to control future transitions.

⋆ The author is partially supported by LATNA Laboratory, NRU HSE, RF government grant, ag. 11.G34.31.0057.

C.A. Coello Coello et al. (Eds.): PPSN 2012, Part II, LNCS 7492, pp. 277–286, 2012.

Consider a general formulation of a combinatorial optimization problem with binary variables:

$$\min\{f(x)|x \in D \subset B^n\}, \tag{1}$$

where B^n is a set of all n-dimensional vectors, whose components are either 1 or 0, and $f(x)$ is an objective function. At each stage of the simulated annealing method, a set of solutions $N(x) \in D$ that belong to a user-defined neighborhood of x is generated according to some predefined rule. The method sequentially evaluates $N(x)$, and moves from the current solution to one of the solutions in $N(x)$ based on the Metropolis acceptance criterion [3], which describes the change of states in thermodynamic systems. The transition from x to $y \in N(x)$ happens with the probability $P(x \to y)$, which depends on a temperature parameter μ:

$$P(x \to y) = \begin{cases} \exp(-\mu[f(y) - f(x)]), & \text{if } f(x) \le f(y); \\ 1 & \text{if } f(x) > f(y). \end{cases}$$

Given a sufficient number of iterations with constant temperature parameter μ, the SA method will converge to an equilibrium state. If we decide to terminate it after reaching the equilibrium, then the final solution – which can be modeled by a random vector $\xi(\mu, \omega)$ – will follow a Boltzmann distribution [4]:

$$P\{\xi(\mu,\omega) = x\} = \begin{cases} \dfrac{\exp(-\mu f(x))}{\sum\limits_{x \in D} \exp(-\mu f(x))}, & x \in D \\ 0, & x \notin D. \end{cases} \tag{2}$$

The set of feasible solutions D can be represented as a union of two disjoint subsets: $D_j^1 = \{x \mid x \in D,\ x_j = 1\}$, and $D_j^0 = \{x \mid x \in D,\ x_j = 0\}$, representing feasible solutions with the j-th component equal to 1 or 0 respectively. The probability that the j-th component of random vector $\xi(\mu)$ is equal to 1 can be expressed as

$$\pi_j(\mu) \equiv P\{\xi_j(\mu) = 1\} = \frac{\sum_{x \in D_j^1} \exp(-\mu f(x))}{\sum_{x \in D} \exp(-\mu f(x))}. \tag{3}$$

The stationary probabilities $\pi_j(\mu)$ can be used to generate random vectors that have approximately Boltzmann distribution provided by (2). Usually, $\xi_j(\mu)$ and $\xi_i(\mu)$ are not independent for all $i \neq j$, therefore we can only achieve an approximation of (2) when generating random solutions by fixing their components independently according to (3). GES collects information about some solutions from D and uses it to approximate the equilibrium distribution (2).

Let S be a subset of D; $S_j^1 = \{x \mid x \in S, x_j = 1\}$, and $S_j^0 = \{x \mid x \in S, x_j = 0\}$. For example, this set can contain all local optima discovered in the past search stages. Instead of explicitly storing the solutions in S, it is sufficient to store only the values that are required to approximate (3). Specifically, we set:

$$Z(\mu) = \sum_{x \in S} \exp(-\mu[f(x) - \min_{x \in S} f(x)]) \tag{4}$$

$$Z_j^1(\mu) = \sum_{x \in S_j^1} \exp(-\mu[f(x) - \min_{x \in S} f(x)]) \tag{5}$$

Every time a new solution is included in S, these memory structures can be updated by adding the corresponding terms to (4) and (5), without storing complete solutions with all attributes. These values are scaled using the value of the best objective function in S, $\min_{x \in S} f(x)$. Whenever the best objective is improved by a newly found solution, $Z(\mu)$ and $Z_j^1(\mu)$ are rescaled by multiplying each value by $\exp(x^{oldbest} - x^{newbest})$, where $x^{oldbest}$ and $x^{newbest}$ are the old best objective value and the new best objective value, respectively.

Using these memory structures, the stationary probabilities $\pi_j(\mu)$ from (3) can be approximated as

$$p_j(\mu) = \frac{Z_j^1(\mu)}{Z(\mu)}. \tag{6}$$

1.1 Intensification and Diversification

GES generates solutions according to the distribution defined by (6), or applies a sequence of perturbations to the components of a given solution guided by (6), which is usually more efficient in practice. The latter approach to generating new solutions should be used when a single perturbation might lead to an infeasible solution, in which case such perturbations are simply prohibited.

By increasing the value of the temperature parameter we can generate solutions that more closely resemble the best solution in the set of known solutions S. Let $x^{min} \in S$ denote the best solution in S: $f(x^{min}) = \min_{x \in S} f(x)$. If all other solutions in S have larger objective functions, then

$$\lim_{\mu \to \infty} p_j(\mu) = x_j^{min}.$$

This follows from the definition of $p_j(\mu)$ and the fact that $\lim_{\mu \to \infty} Z(\mu) = 1$. This property is used in GES to alternate between diversification and intensification stages using a monotonically increasing sequence of temperature parameters: $\mu_1, \mu_2, \ldots, \mu_K$. In order to calculate $p_j(\mu_k)$ for all temperature values from this sequence, we need to store $(n+1) \cdot K$ values corresponding to $Z_j^1(\mu_k)$ and $Z(\mu_k)$, where n is the number of binary variables in the problem definition.

The specific values for the temperature parameters are usually calculated using simple recursive formulas: $\mu_0 = 0$, $\mu_{k+1} = \alpha \mu_k$ for $k = 1, \ldots, K-1$. The parameters involved in this recursion (μ_1, α and K) are chosen to guarantee the convergence to the best solution in the set S:

$$\| x_j^{min} - p_j^K \| \approx 0$$

Below, we will address the dynamic adjustment of the temperature schedule that can be easily implemented in practice.

1.2 Alternative Memory Implementations

Even for small values of the temperature parameter μ, the expression given by (6) can provide biased probabilities. For example, if we generate solutions with temperature equal to zero and the set S contains only solutions with jth component equal to 1, then the probability $p_j(0) = 1$. To deal with this bias, one can use alternative memory implementations.

For example, the probabilities can be approximated using the best objective values corresponding to each component:

$$f_j^0 = \begin{cases} \min\{f(x) : x \in S_j^0\} & \text{if } |S_j^0| \neq 0 \\ \infty & \text{otherwise} \end{cases}$$

$$f_j^1 = \begin{cases} \min\{f(x) : x \in S_j^1\} & \text{if } |S_j^1| \neq 0 \\ \infty & \text{otherwise} \end{cases}$$

Let x^{min} be a solution from the set S with a minimum value: $f(x^{min}) = \min\{f(x) : x \in S\}$. The probability $p_j(\mu)$ can be approximated as:

$$p_j(\mu) = \frac{\exp(-\mu[f_j^1 - f(x^{min})])}{\exp(-\mu[f_j^1 - f(x^{min})]) + \exp(-\mu[f_j^0 - f(x^{min})])} \tag{7}$$

In Figure 1, we present a pseudo-code of the procedure that calculates the generation probabilities using f_j^1 and f_j^0. To achieve convergence to the best solution in the set S when using Formula (7), we penalize solutions that have the same objective value as x^{min} (Figure 1, lines 10–13; 20–23). In addition, if S_j^1 (or S_j^0) is empty, one can use the maximum absolute difference between objective values instead of $f_j^1 - f(x^{min})$ ($f_j^0 - f(x^{min})$) to avoid premature convergence (Figure 1, lines 5, 15).

The general scheme of the GES method is presented in Figure 2. In the beginning memory structures are initialized by a randomly generated solution. The temperature cycle is repeated until $nfail^{max}$ cycles without improvement to the best found solution, x^{best}. Generation probabilities are recalculated at every temperature stage using the corresponding temperature parameter. These probabilities are used to generate $ngen$ solutions that are used as initial points for local search procedure. Locally-optimal solutions are used to update the adaptive memory structures, which will affect the future generation probabilities.

1.3 Parallel Implementations

GES algorithm can be naturally extended for parallel computing as it performs search simultaneously in distinct areas of the solution space. One can trivially accelerate GES by initiating a set of copies of GES procedures with different random seeds. Due to the randomness, each copy will follow a distinct search trajectory, which often leads to significant parallel acceleration in practice [5]. Further

Require: μ – temperature parameter μ, x^{min} – current best solution, f^0 and f^1 – vectors of the best object values corresponding to each component, n – number of solution components;

1: $maxdif = \max_j\{abs(f_j^0 - f_j^1) : f_j^1 < \infty; f_j^0 < \infty;\}$
2: $sum_0 = sum_1 = 0$
3: **for** $j = 1$ to n **do**
4: **if** $f_j^0 = \infty$ **then**
5: $sum_0 = sum_0 + \exp(-\mu \cdot maxdif)$
6: **else**
7: **if** $f_j^0 > f(x^{min})$ **then**
8: $sum_0 = sum_0 + \exp(-\mu[f_j^0 - f(x^{min})])$
9: **else**
10: **if** $x_j^{min} = 0$ **then**
11: $sum_0 = sum_0 + 1$
12: **else**
13: $sum_0 = sum_0 + \exp(-\mu)$
14: **if** $f_j^1 = \infty$ **then**
15: $sum_1 = sum_1 + \exp(-\mu \cdot maxdif)$
16: **else**
17: **if** $f_j^1 > f(x^{min})$ **then**
18: $sum_1 = sum_1 + \exp(-\mu[f_j^1 - f(x^{min})])$
19: **else**
20: **if** $x_j^{min} = 1$ **then**
21: $sum_1 = sum_1 + 1$
22: **else**
23: $sum_1 = sum_1 + \exp(-\mu)$
24: $p_i(\mu) = \frac{sum_1}{sum_1 + sum_0}$
25: **return** $p(\mu)$

Fig. 1. Calculation of the transition probabilities

improvements can be achieved by sharing the best found solutions and/or the adaptive memory structures (for example, by sending the updates to vectors f^0 and f_j^1) between different copies. Such sharing is equivalent to increasing the number of solutions, *ngen*, generated at each temperature stage of GES algorithm.

2 Applications

Global equilibrium search has a number of advantages when compared to the simulated annealing method. Firstly, the presence of adaptive memory allows GES to outperform SA in terms of solution quality and computational speed. Its performance was tested on classic optimization problems that capture the complexities of modern optimization applications. Here we mention some of these applications and provide detailed results for the quadratic assignment problem.

Require: μ – vector of temperature values, K – number of temperature stages, $maxnfail$ – restart parameter, $ngen$ – # of solutions generated during each stage
Ensure:
1: x^{best} = construct random solution; S=$\{x^{best}\}$
2: **while** stopping criterion = FALSE **do**
3: $x \leftarrow$ construct random solution
4: $x^{min} = x$
5: $S = \{x^{min}\}$ (set of known solutions)
6: **for** $nfail = 0$ to $nfail^{max}$ **do**
7: $x^{old} = x^{min}$
8: **for** $k = 0$ to K **do**
9: $p(\mu_k)$ = calculate generation probabilities(S, μ_k)
10: **for** $g = 0$ to $ngen$ **do**
11: x = generate solution$(x, p(\mu_k))$
12: x = local search method(x)
13: $S = S \cup x$
14: $x^{min} = \arg\min\{f(x) : x \in S\}$
15: **if** $f(x^{min}) < f(x^{best})$ **then**
16: $x^{best} = x^{min}$
17: **if** $f(x^{old}) > f(x^{min})$ **then**
18: $nfail = 0$
19: **return** x^{best}

Fig. 2. Pseudo-code of the GES method

2.1 Unconstrained Binary Quadratic Problem

One of the well-known and most interesting classes of integer optimization problems is the maximization of the quadratic 0–1 function:

$$\max_{\mathbf{x}\in\{0,1\}^n} f(\mathbf{x}) = \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}x_i x_j, \tag{8}$$

where q_{ij} are elements of an $n \times n$ symmetric real matrix $Q \in \mathbb{R}^{n\times n}$. This problem is referred to as an unconstrained binary quadratic programming problem. Many fundamental problems in science, engineering, finance, medicine and other diverse areas can be formulated as quadratic binary programming problems. Quadratic functions with binary variables naturally arise in modeling selections and interactions.

GES was applied to a wide spectrum of large-scale instances of the unconstrained binary quadratic programming problem [6]. The computational experiments revealed favorable performance compared to the best known heuristics on a set of publicly available benchmark instances [7,8,9].

2.2 Maximum Satisfiability Problem

Maximum satisfiability problem consists of finding an assignment of boolean variables that satisfies as many given logical clauses as possible. In the weighted

maximum satisfiability problem each logical clause has a predetermined positive weight, and the goal is to search for an assignment, which maximizes the total weight of the satisfied clauses.

Among various heuristic approaches for solving this problem – such as, algorithms based on reactive tabu search [10], simulated annealing [11], GRASP [12,13,14,15,16], iterated local search [17] and guided local search [18] – GES provides the state-of-the-art performance [19] on many benchmark instances [20,21,22,23].

2.3 Quadratic Assignment Problem

The Quadratic Assignment Problem (QAP) stated for the first time by Kooper and Beckman in 1957 is a well known combinatorial optimization problem and remains one of the greatest challenges in the field [24], [25]. Firstly it was formulated in the context of the plant location problem. Given n facilities with some physical products flow between them and n locations with known pairwise distances, one should determine to which location each facility must be assigned in order to minimize total distance $\times$ flow.

Mathematically QAP can be formulated as follows: let $A_{n\times n} = a(i,j)$ be a matrix, where $a_{i,j}R^+$ represents product flow between facilities f_i and f_j, let $B_{n\times n} = (b_{i,j})$ be a matrix, where $b_{i,j} \in R^+$ represents the distance between locations l_i and l_j. Let $p : \{1,\ldots,n\} \to \{1,\ldots,n\}$ be a permutation of integers. The cost of a permutation is defined as follows:

$$c(p) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}b_{p(i)p(j)}.$$

The goal is to find a permutation $p*$ with a minimal cost. The QAP is well known to be strongly NP-hard, and even small instances may require long computational time. A number of practical problems can be formulated as QAP. Among those are problems dealing with backboard wiring, scheduling, manufacturing, statistical data analysis, typewriter keyboard design, image processing, turbine balancing and so on [25], [26].

We performed a series of computational experiments on well known benchmark problems from the QAPLIB library [27] (online version is available at http://www.opt.math.tu-graz.ac.at/qaplib/). In our implementation we used Tabu Search algorithm as a local search method [28]. The neighborhood of a given permutation $\hat{p}$ was defined as $N(\hat{p}) = \{p : \|p - \hat{p}\| = 2\}$, where $\|\cdot\|$ denotes Hamming distance.

We compared our algorithm to Robust Tabu search (RoTabu), Ant Colony and Simulated Annealing (SA) algorithms (the codes for these algorithms were obtained from QAPLIB resource as well). Each algorithm was executed on each instance 10 times. The experiments were performed on a 3GHz AMD computer. In Table 1 and Table 2 we report the average deviation (in %) between the best solution found by the algorithm and the best known solution.

Table 1. Experiments on real world instances. *Instance* - problem instance name, *n* - size of the instance, *BKS* - best known solution value, *time* - maximum allowed computational time in seconds.

Instance	n	BKS	RoTabu	SA	Ant Colony	GES	time, s
bur26a	26	5426670	0.03	0.52	**0.02**	0.03	0.1
bur26b	26	3817852	0.09	0.32	**0.03**	0.07	0.1
bur26c	26	5426795	0.03	0.29	**0**	**0**	0.1
bur26d	26	3821225	0.05	0.10	**0**	**0**	0.1
bur26e	26	5386879	0.01	0.18	**0**	**0**	0.1
bur26f	26	3782044	0.01	0.06	**0**	**0**	0.1
bur26g	26	10117172	0.01	0.30	**0**	**0**	0.1
bur26h	26	7098658	0.13	0.13	**0**	**0**	0.1
chr25a	25	3796	4.23	45.5	1.24	**0**	2
nug30	30	6124	0.04	5.29	0.15	**0**	2
kra30a	30	88900	**0**	3.77	0.70	**0**	4
kra30b	30	91420	**0.01**	3.59	0.04	**0.01**	4
tai64c	64	1855928	0.37	1.28	0	**0**	1
tai20b	20	122455319	0.05	8.75	0.09	**0**	0.1
tai25b	25	344355646	0.02	2.82	0	**0**	0.5
tai30b	30	637117113	0.04	2.66	0	**0**	1
tai35b	35	283315445	0.1	3.09	0	**0**	2
tai40b	40	637117113	0.43	2.12	0.11	**0**	2
tai50b	50	458821517	1.58	0.57	0.26	**0**	8
tai60b	60	608215054	1.05	0.66	0.32	**0**	20
tai80b	80	818415043	0.84	1.43	0.94	**0.12**	40

Table 2. Experiments on randomly generated instances. *Instance* - problem instance name, *n* - size of the instance, *BKS* - best known solution value, *time* - maximum allowed computational time in seconds.

Instance	n	BKS	RoTabu	SA	Ant Colony	GES	time, s
tai20a	20	703482	**0.05**	0.55	0.44	0.06	2.5
tai25a	25	1167256	**0**	0.93	1.5	**0**	5
tai30a	30	1818146	0.29	0.47	0.93	**0.03**	7.5
tai35a	35	2422002	0.63	0.86	1.14	**0.16**	10
tai40a	40	3139370	0.78	1.01	1.43	**0.30**	30
tai50a	50	4941410	1.04	1.37	1.75	**0.64**	45
tai60a	60	7205962	1.17	1.25	1.94	**0.84**	60
tai80a	80	13546960	1.28	1.07	1.39	**0.62**	120

3 Conclusions

One of the most important qualities of GES is its ability to process and utilize the solutions that are obtained by different search techniques. GES offers a mechanism of information processing that can be used to organize an intelligent

multi-start search that involve different optimization techniques. This collaborative functionality can be effectively used in parallel implementations. Numerous successful applications on a wide range of combinatorial optimization problems corroborate the efficiency of the GES method.

References

1. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science 220(4598), 671–680 (1983)
2. Shylo, V.: A global equilibrium search method (in russian). Kybernetika i Systemniy Analys 1, 74–80 (1999)
3. Schuur, P.C.: Classification of acceptance criteria for the simulated annealing algorithm. Mathematics of Operations Research 22(2), 266–275 (1997)
4. Henderson, D., Jacobson, S., Johnson, A.: The theory and practice of simulated annealing. In: Glover, F., Kochenberger, G. (eds.) Handbook of Metaheuristics, pp. 287–319. Kluwer Academic Publishers, Boston (2003)
5. Shylo, O.V., Middelkoop, T., Pardalos, P.M.: Restart strategies in optimization: parallel and serial cases. Parallel Comput. 37, 60–68 (2011)
6. Pardalos, P.M., Prokopyev, O.A., Shylo, O.V., Shylo, V.P.: Global equilibrium search applied to the unconstrained binary quadratic optimization problem. Optimization Methods Software 23(1), 129–140 (2008)
7. Palubeckis, G.: Unconstrained binary quadratic optimization, `http://www.soften.ktu.lt/~gintaras/binqopt.html` (last accessed on October 2008)
8. Beasley, J.E.: OR-library webpage, `http://people.brunel.ac.uk/ mastjjb/jeb/info.html` (last updated on June 2008)
9. Sloane, N.: Challenge problems: Independent sets in graphs, `http://www.research.att.com/~njas/doc/graphs.html` (last accessed on October 2008)
10. Battiti, R., Protasi, M.: Reactive search, a history-sensitive heuristic for max-sat. J. Exp. Algorithmics 2 (January 1997)
11. Spears, W.M.: Simulated annealing for hard satisfiability problems. In: Johnson, D., Trick, M. (eds.) The Second DIMACS Implementation Challenge. DIMACS Series on Discrete Mathematics and Theoretical Computer Science, vol. 26, pp. 533–558. American Mathematical Society (1996)
12. Du, D., Gu, J., Pardalos, P.: Satisfiability problem: theory and applications: DIMACS Workshop, March 11-13, vol. 35. Amer. Mathematical Society (1997)
13. Festa, P., Pardalos, P.M., Pitsoulis, L.S., Resende, M.G.C.: Grasp with path relinking for the weighted maxsat problem. J. Exp. Algorithmics 11, Article No. 2.4 (2006)
14. Festa, P., Pardalos, P.M., Pitsoulis, L.S., Resende, M.G.C.: Grasp with path relinking for the weighted maxsat problem. J. Exp. Algorithmics 11 (February 2007)
15. Resende, M., Feo, T.: A grasp for satisfiability. In: Johnson, D., Trick, M. (eds.) The Second DIMACS Implementation Challenge. DIMACS Series on Discrete Mathematics and Theoretical Computer Science, vol. 26, pp. 499–520. American Mathematical Society (1996)
16. Resende, M., Pitsoulis, L., Pardalos, P.: Approximate solution of weighted max-sat problems using grasp. In: Gu, J., Pardalos, P. (eds.) Satisfiability Problem: Theory and Applications. DIMACS Series on Discrete Mathematics and Theoretical Computer Science, vol. 35, pp. 393–405. American Mathematical Society (1997)

17. Yagiura, M., Ibaraki, T.: Analyses on the 2 and 3-flip neighborhoods for the max sat. Journal of Combinatorial Optimization 3, 95–114 (1999)
18. Mills, P., Tsang, E.: Guided local search for solving sat and weighted max-sat problems. J. Autom. Reason. 24(1-2), 205–223 (2000)
19. Shylo, O.V., Prokopyev, O.A., Shylo, V.: Solving weighted max-sat via global equilibrium search. Oper. Res. Lett., 434–438 (2008)
20. Resende, M.: Algorithm source code distribution, `http://research.att.com/~mgcr/src` (last accessed on October 2008)
21. Resende, M.: Test problem distribution, `http://research.att.com/~mgcr/data` (last accessed on October 2008)
22. Yagiura, M.: Benchmark instances and program codes for max sat, `http://www.al.cm.is.nagoya-u.ac.jp/~yagiura/msat.html` (last accessed on October 2008)
23. Mills, P.: Constraint programming and optimization laboratory, university of essex, gls solver, algorithm: Extended gls, `http://cswww.essex.ac.uk/CSP/glsdemo.html` (last accessed on October 2008)
24. Pardalos, P., Wolkowicz, H.: Quadratic assignment and related problems: DIMACS Workshop, May 20-21, vol. 16. Amer. Mathematical Society (1994)
25. Burkard, R., Cela, E., Pitsoulis, L., Pardalos, P.M.: The quadratic assignment problem. In: Handbook of Combinatorial Optimization, vol. 3, pp. 241–337. Springer (1998)
26. Pardalos, P., Pitsoulis, L.: Nonlinear assignment problems: algorithms and applications, vol. 7. Kluwer Academic Pub. (2000)
27. Burkard, R.E., Karisch, S.E., Rendl, F.: Qaplib–a quadratic assignment problem library. Journal of Global Optimization 10(4), 391–403 (1997)
28. Misevicius, A.: A tabu search algorithm for the quadratic assignment problem. Computational Optimization and Applications 30(1), 95–111 (2005)