OR Spektrum (1995) 17:205–210

OR Spektrum

A modification of threshold accepting and its application to the quadratic assignment problem

Volker Nissen, Henrik Paul

Institut für Wirtschaftsinformatik, Abt.I, Universität Göttingen, Platz der Göttinger Sieben 5 (MZG), D-37073 Göttingen, Germany

Received: 27 August 1993 / Accepted in revised form: 1 July 1994

Abstract. In this paper we propose a modification of the threshold accepting heuristic by Dueck and Scheuer. Instead of using discrete threshold values a threshold function similar to the cooling schedule of simulated annealing is used. Furthermore, the number of iterations during each step of the heuristic is a function of the current and the initial threshold value. Using this scheme, we investigate the trade-off between solution quality and convergence speed on different instances of the well known quadratic assignment problem. In a second set of experiments the results of a multistart-version of TA are compared with the results of unique long runs at identical CPU-requirements to identify the better optimization strategy. Since, generally, in the literature the number of starting solutions for QAP-heuristics appears to be chosen on a rather arbitrary basis, we also highlight how varying this number influences the TA-results.

Zusammenfassung. Im vorliegenden Beitrag wird eine Modifizierung der Threshold Accepting Heuristik von Dueck und Scheuer vorgeschlagen. Anstelle diskreter Schwellenwerte wird eine Schwellenwertfunktion verwendet, die vom Abkühlungsplan beim Simulated Annealing inspiriert ist. Desweiteren ist die Iterationszahl auf jeder Ebene des Verfahrens nunmehr eine Funktion des aktuellen sowie des Ausgangsschwellenwertes. Anhand dieses Vorgehensschemas untersuchen wir den Trade-off von Lösungsqualität und Konvergenzgeschwindigkeit bei verschiedenen Standardbeispielen des bekannten Quadratischen Zuordnungsproblems. Auch die Qualität und Zuverlässigkeit einer Multistart-Version kurzer TA-Läufe wird mit den Ergebnissen ausführlicher Läufe bei gleichen CPU-Zeiten verglichen, um Rückschlüsse auf die sinnvollere Optimierungsstrategie zu erhalten. In der Literatur verwenden unterschiedliche Autoren häufig sehr verschiedene Anzahlen zufälliger Startlösungen in ihren numerischen Experimenten. Wir untersuchen daher auch, wie sich eine Variation dieser Anzahl auf die TA-Ergebnisse auswirkt.

Key words: Optimization, local search, heuristic, threshold accepting, quadratic assignment problem

Schlüsselwörter: Optimierung, lokale Suchverfahren, Heuristik, Threshold Accepting, Quadratisches Zuordnungsproblem

Correspondence to: V. Nissen

1. Introduction

Threshold accepting (TA) is a simplification of the well known simulated annealing (SA) method [10] that was proposed by Dueck and Scheuer [4]. TA is a local search method. Starting from an initial solution each TA-step consists of a slight change of the old solution in a new one. Then the qualities of the two solutions are compared with respect to the given objective function. TA accepts any new solution that is not much worse than the old one. More specifically, TA accepts every solution that is either better than the current solution or that deteriorates the old objective function value by less than a given threshold level T. The new solution then replaces the current old solution as a basis for the next TA-step. The threshold T will be relatively large in the beginning of the search process to allow for a full exploration of the solution space. As the search continues, T is lowered in a stepwise manner. Generally, an incrasing number of trials is performed at successive levels since lower thresholds will expand the time required to reach some form of equilibrium or ground state. The search process terminates when a minimum threshold level is reached or some other termination criterion holds.

We apply a modified TA to the quadratic assignment problem (QAP) - one of the most difficult problems in combinatorial optimization. We first describe our modified version of TA in the following section before defining the QAP formally in Sect. 3. Then, empirical results for a test suite of QAPs with very different size and structure are given. Emphasis is on the trade-off between solution quality and convergence speed when exogeneous strategy parameters of the heuristic are being changed. We also investigate the influence of varying the number of random starting solutions on the TA-statistics, and compare the performance of a TA multistart-version to the results achieved with unique long runs at approx. identical CPU-requirements. The last section summarizes our findings.

2. A modified TA-scheme

In their implementation Dueck and Scheuer use a prespecified vector of threshold values. This is a rather inflexible method that is particulary inconvenient during the test phase on a given problem. Moreover, extra tuning of the threshold vector seems necessary when the heuristic is applied to different problem classes.

We propose to use the following threshold function instead. It resembles the cooling schedule of SA:

$$T_t = \begin{cases} T_0 & t = 1 \\ \alpha \cdot T_{t-1} & t = 2, \ldots \end{cases} \quad (1)$$

where T_0 : initial threshold value
T_t : threshold value at level t
α : lowering factor α=]0,1[.

The factor α is a stategy parameter of our TA implementation that should be set to values between 0.8 and 0.995. The initial threshold value T_0 may be calculated from a random sample of m solution trials as

$$T_0 = \beta \cdot \frac{1}{m} \sum_{i=1}^{m} f_i \quad (2)$$

where f_i : objective function value of trial i
β : constant scalar
(β = 0.05, m = 3 in all experiments).

For each level of the TA search process the number of iterations I_t (i.e. evaluated trial solutions) must be determined. Since locating improved solutions and leaving suboptima becomes increasingly difficult with time, I_t should be higher in later search phases than in the beginning. The following iteration function is used:

$$I_t = \begin{cases} \frac{c \cdot T_0 \cdot I_{\max}}{T_t} & c \cdot T_0 \leq T_t \leq T_0 \\ I_{\max} & \text{else} \end{cases} \quad (3)$$

where I_t : iterations to be performed at level t
$I_{\max}$: maximum number of iterations (strategy parameter)
c : weighting factor
(c = 0.1 in all experiments).

With this function the number of iterations at a given level of TA depends on the current threshold value T_t as well as on the initial T_0. A maximum number of iterations $I_{\max}$ was set to avoid excessive CPU-requirements. Finally, a new solution is obtained from a given configuration by a simple two-exchange (swap) of solution elements. In our implementation, the solution elements to be swapped are randomly determined to minimize stagnation due to circular swapping. The TA search terminates when either T_t has fallen below some minimal threshold level $T_{\min}$ = 1, the complete neighborhood (points reachable by a simple swap) of a given solution has been unsuccessfully evaluated, or no improvement has been achieved for ten successive levels. Note that in the second case a further lowering of the threshold would be pointless since no improvements are possible. The third termination criterion prevents excessive calculations if circular swapping occurs. An overview of the modified TA implementation is given in Fig. 1.

```
10   Start
20   Choose α, β, c and I_max as strategy parameters
30   Generate initial threshold T_0
40   Determine initial solution and save as current best_solution
50   t = 0
60   Repeat
70       t = t + 1
80       calculate T_t and I_t
90       Iteration_count = 0
100      Repeat
110          Iteration_count = Iteration_count + 1
120          Generate new solution by random two-exchange
130          Calculate difference Δz between objective function
             values
               (new solution - old solution)
140          If Δz < T_t then accept new solution
150          If new solution better than best_solution
               then update best_solution
160      until (Iteration_count = I_t) or (entire neighborhood searched
         without success)
170  until (T_t < T_min) or (entire neighborhood searched
     without success) or (no improvement for ten successive levels)
180  print best_solution
190  Stop
```

Fig. 1. Modified threshold accepting in pseudo code (minimization)

3. The quadratic assignment problem (QAP)

Locating facilities with material flow between them is a difficult layout problem. Koopmans and Beckmann [11] were the first to model this in a way which is now known as the quadratic assignment problem. Formally, it can be stated as a permutation problem [2]. Given a set N≡{1,2,...,n} and real numbers c_{ik}, a_{ik}, b_{ik} for i,k= 1,2,...,n, find a permutation φ of the set N which minimizes:

$$Z = \sum_{i=1}^{n} c_{i\varphi(i)} + \sum_{i=1}^{n} \sum_{k=1}^{n} a_{ik} b_{\varphi(i),\varphi(k)} \; . \quad (4)$$

The linear part may be viewed as installation costs. The quadratic part accounts for interaction between facilities, summing the product of material flow between each two facilities and the cost of transferring a material unit between their respective locations.

Vollmann and Buffa [20] introduced the concept of flow-dominance as a measure of the variation of values in the flow matrix. It is given by 100 × std.dev./mean of the matrix elements. Simply stated, high flow-dominance indicates that few facilities with high interaction tend to dominate the problem.

As a generalization of the TSP the QAP is NP-hard. Sahni and Gonzales showed that even finding an ϵ-approximation for the QAP is NP-hard. Many solution methods have been developed to address the QAP because of its considerable practical importance in facility layout, machine scheduling and other areas of application. Even with large computers, exact (implicit enumeration) algorithms such as branch-and-bound methods [6] are only able to globally solve rather small QAPs (n≈18) in a reasonable amount of time. Therefore, researchers have concentrated on developing effective heuristics for the QAP.

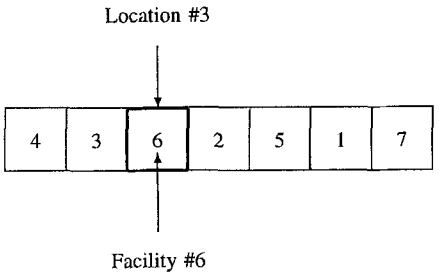

Fig. 2. Representation of a QAP-solution in TA (n=7)

The diverse QAP-heuristics are not reviewed here. Extensive reviews of the QAP and associated solution techniques can be found in [2, 13]. Recent developments in facility layout are also covered in [8].

4. Empirical results with TA

The modified TA-scheme was tested on a number of well known problems taken from [14] (NUG15, NUG20, NUG30), [18] (STE36a, STE36c), [5] (ELS19), [12] (KRA30a, KRA30b) and [17] (SKO64, SKO81, SKO90). The testproblems vary in size between 15 and 90 facilities. The instances by Nugent et al. and Skorin-Kapov are randomly generated with low flow-dominance. The other appear to be practical applications with higher flow-dominance values. Hence the problems are very different in size and structure making them a good test suite. These and other problem instances are collected in the QAPLIB problem library described in [3].

The solution representation of our TA is a straightforward permutation coding (see Fig. 2). Each position on a solution vector represents a facility location. Integer values are assigned as numbered facilities.

Three different settings for the exogeneous parameters α and I_{max} are used, specifying threshold and iteration function respectively. These settings are given in Table 1. Strategy one (TA I) aims at achieving high quality solutions, allowing a rather intensive search. Strategy three (TA III) is a quick optimizer, but sacrifices some of the attainable solution quality. Strategy two (TA II) is a compromise between the other two.

All parameters are determined on the basis of some pretests but no extensive testing was done, and, thus, no claim is made as to their optimality. In a hybridized TA version we also employ 2-Opt to finally improve the generated TA solution. 2-Opt is a simple local search heuristic that sequentially considers pairwise exchange between the positions of facilities. The swap is made whenever it leads to a lower objective function value and then the search starts again from the new solution. This procedure continues until no exchange in the current solution results in a further improvement. Ten trials are performed on each problem instance in all experiments. Results are given in Table 2.

To give an idea of the quality of TA-results we present data from the well-known Tabu Search implementation of Taillard in Table 3. Since different hardware and programming languages are employed it is more appropriate to relate solution quality to function evaluations than to CPU-requirements when comparing TA and TS. One should, however, be careful to note that TS may be implemented in such a way [19, 21] that it has a lower time complexity to evaluate the quality of a single swap than has TA. In other words, a given number of function evaluations requires less time for TS than it does for TA on identical hardware.

Table 1. Three different parameter settings used in our TA-implementation for the QAP. The first strategy optimizes quality, the third convergence speed while the second strategy is a compromise between these two

Problem	Strat.1		Strat.2		Strat.3	
	I_{max}	α	I_{max}	α	I_{max}	α
NUG15	250	0,995	190	0,975	130	0,8
NUG20	280	0,995	210	0,975	140	0,8
NUG30	420	0,995	310	0,975	200	0,8
ELS19	300	0,995	230	0,975	150	0,8
KRA30a	400	0,995	300	0,975	200	0,8
KRA30b	400	0,995	300	0,975	200	0,8
STE36a	800	0,995	600	0,975	400	0,8
STE36b	800	0,995	600	0,975	400	0,8
SKO64	700	0,995	530	0,975	350	0,8
SKO81	900	0,995	650	0,975	450	0,8
SKO90	1000	0,995	750	0,975	500	0,8

The three TA-strategies demonstrate expected behavior. A slower lowering of the threshold-value combined with more iterations at each level allows for more intensive search and better chances to escape from local optima. This leads to increased solution quality and generally more reliable optimization (lower standard deviations) at the expense of more function evaluations. However, it is noteworthy that even the very fast schedule of TA III produces solutions of reasonable quality. TA III is a good choice when optimization must proceed rapidly – as in real-time applications – or when each function evaluation is time-consuming or costly. Adding a 2-Opt post-processing may slightly improve the TA III-result at very low cost.

TA I and TA II tend to produce locally optimal results so that an additional 2-Opt becomes rather superfluous. The average quality of TA I solutions is always better than for TA II. TA I is an efficient QAP-heuristic that produces good results on a suite of test problems with very different size and structure. It is particularly strong on the large problem instances. Not surprisingly, the TA performance is better on problems with low flow-dominance. High flow-dominance values tend to make it more difficult for a 2-swap based heuristic to escape from local optima. It should be noted that an even slower lowering of the TA threshold and/or a larger value for I_{max} can be expected to further increase the solution quality.

Our TA implementation uses only one problem-specific parameter (I_{max}) and may directly be applied to other combinatorial problems, possibly allowing for a change in the employed search operator (neighborhood structure). This makes the modified TA an efficient, easily implementable and user-friendly heuristic of broad applicability. It also has moderate CPU-requirements on a serial computer and may be tuned for either speed or solution quality or any compromise between these objectives.

We were wondering whether, given a fixed amount of CPU-time, it would be more successful to conduct some long

Table 2. Empirical results with the three different TA-strategies. Results averaged over 10 runs on a DEC-α-workstation. The starting solutions were randomly generated and identical for all TA-versions

	Average objective function values and function evaluations					
Testproblem	TA I (+ 2-Opt)	AFE	TA II (+ 2-Opt)	AFE	TA III (+ 2-Opt)	AFE
NUG15	1,156.2	29,863.1	1,160.6	6,602.7	1,172.2	1,164.1
	1,155.0	29,971.1	1,160.6	6,707.7	1,172.2	1,269.1
NUG20	2,584.8	72,001.9	2,604.2	17,787.0	2,649.0	1,876.1
	2,583.6	72,224.1	2,603.4	17,986.6	2,647.8	2,088.7
NUG30	6,148.4	225,157.1	6,178.6	35,997.4	6,330.0	3,457.6
	6,147.6	225,657.5	6,178.6	36,432.4	6,304.4	4,293.1
ELS19	18,684,375.0	269,592.0	21,404,362.4	52,478.2	21,593,135.0	6,036.4
	18,684,139.2	269,770.9	21,401,541.6	52,678.6	21,593,135.0	6,207.4
KRA30a	90,109.0	180,220.3	91.171.0	30,639.4	93,632.0	5,400.3
	90.109.0	180,655.3	91,171.0	31,074.4	93,632.0	5,835.3
KRA30b	92,019.0	179,316.2	93,019.0	31,620.0	93,998.0	5,510.1
	91,999.0	179,785.5	93,019.0	32,055.0	93,998.0	5,945.1
STE36a	9,614.6	516,420.7	9,833.6	97,359.1	10,245.2	8,854.0
	9,605.8	517,115.5	9,831.6	97,985.6	10,230.4	9,722.1
STE36b	16,057.2	574,154.8	16,738.2	103,081.8	17,245.0	11,321.5
	16,057.2	574,784.8	16,727.4	103,742.1	17,245.0	11,951.5
SKO64	48,747.0	764,803.3	48,919.2	121,263.2	49,754.0	9,206.2
	48,745.6	767,236.2	48,919.0	123,451.4	49,308.6	38,226.8
SKO81	91,299.6	1,096,761.4	91,662.0	163,972.7	93,552.0	13,184.7
SKO90	115,981.2	1,210,810.2	116,418.8	194,475.6	118,613.2	15,121.1

AFE = average function evaluations

Table 3. Numerical results of the TS-implementation by Taillard [19]. According to personal communication with E. Taillard, the number of function evaluations in one TS iteration is given by $n \cdot (n-1)/2$. TS was run on a 10-node T800C-G20S transputer system. CPU-time per iteration is given by approx. $6,2 \cdot n^2 \mu s$

	Best known	Average objective function value[a] after i TS-iterations					
Testproblem	solution[b]	$i = 4N$	AFE[c]	$i = n^2$	AFE[d]	$i = 1000$	AFE[e]
NUG15	1,150	1,162	6,300	1,152	23,625	1,150	105,000
NUG20	2,570	2,606	15,200	2,580	76,000	2,573	190,000
NUG30	6,124	6,228	52,200	6,148	391,500	6,142	435,000
ELS19	17,212,548	21,154,221	12,996	19,174,778	61,731	17,780,562	171,000
KRA30a	88,900	92,901	52,200	91,567	391,500	91,478	435,000
KRA30b	91,420	94,071	52,200	92,426	391,500	92,334	435,000
STE36a	9,526	10,145	90,720	9,869	816,480	9,917	630,000
STE36b	15,852	18,452	90,720	17,199	816,480	17,310	630,000
SKO64	48,498	49,225	516,096	48,692	8,257,536	48,837	2,016,000
SKO81	91,008	92,100	1,049,760	91,372	21,257,640	91,736	3,240,000
SKO90	115,534	117,036	1,441,800	115,996	32,440,500	116,574	4,005,000

AFE = average function evaluations
[a] The values are approximate. Taillard gives pars per mille above best known solutions
[b] See [19, 21]
[c] $4n \cdot n \cdot (n-1)/2$
[d] $n^2 \cdot n \cdot (n-1)/2$
[e] $1000 \cdot n \cdot (n-1)/2$

runs of TA I or to start the much quicker TA III over and over again from different initial points (TA III multistart-version). Results for a set of four QAPs of different size and structure are given in Table 4. Ten runs are conducted for TA I, while the number of TA III restarts varies with problem size. The total amount of CPU-time for the runs of TA I and multistart TA III respectively is approx. identical. Numerical results clearly indicate that TA I is a more reliable optimizer (lower std. dev.) that achieves much better average performance. The extreme complexity of the QAP search space demands a rather slow cooling schedule to reliably achieve excellent solutions.

One can argue that in practical applications not the average performance but only the best solution found is of importance, since just this solution will be implemented. Focussing solely on the best result, the multistart-version of TA III can be a reasonable choice for small QAPs like ELS19, but, in general, appears to be inferior to TA I with respect to this criterion as well.

Finally, a set of experiments is conducted to investigate how the number of starting solutions (equivalent to test runs)

Table 4. Empirical comparison of TA I and a multistart version of TA III at approx. identical CPU-requirements

Test-problem	TA I				TA III Multistart			
	Best	Worst	Mean	Std.dev.	Best	Worst	Mean	Std.dev.
NUG30	6,124	6,190	6148.4	19.6	6172	6530	6323.0	65.7
ELS19	17,937,024	20,510,786	18,684,375.0	1,143,338.1	17,212,548	28,412,188	21,608,555.3	2,959,917
STE36b	15,852	16,470	16,057.2	191.5	15,852	19,914	17,577.4	801.8
SKO90	115,694	116,242	115,981.2	158.7	117,122	119,816	118,549.8	465.7

Table 5. Varying the number of randomly generated starting solutions (= test runs) for TA I. The resulting differences in QAP-statistics indicate no advantage in choosing a large number of starting solutions to evaluate algorithmic performance

Testproblem	5 starting solutions		10 starting solutions		30 starting solutions	
	Mean	Std.dev.	Mean	Std.dev.	Mean	Std.dev.
NUG30	6,156.8	22.8	6148.4	19.6	6154.9	26.2
ELS19	18,966,057.2	1,260,303.4	18,684,375.0	1,143,338.1	19,198,436.0	1,136,244.3
KRA30a	90,696.0	1,417.2	90,109.0	1,290.6	90,494.7	1,042.3
STE36b	16,050.8	139.0	16,057.2	191.5	16,120.3	261.5
SKO64	48,872.0	260.0	48,747.0	229.6	48,699.1	154.0
SKO90	116,013.2	130.5	115,981.2	158.7	115,902.9	163.1

influences the TA I statistics (mean, std. dev.). In the literature this number seems to be chosen on a rather arbitrary basis when a QAP-heuristic is being evaluated in numerical experiments. Heragu and Alfa [9] conduct five runs, while Burkard and Rendl [1] mostly use ten restarts. Taillard [19] employs 30 different starting solutions, while Paulli [15] even evaluates 100 runs.

Additional runs are costly with respect to CPU-time. Under practical considerations this time might be well invested when one aims at identifying a single good solution while all other results are being discarded. However, if one wishes to investigate the effectiveness and quality of a QAP-heuristic in general, mean performance measures are of interest. We perform experiments using five, ten and 30 random initial solutions for TA I. The results (Table 5) indicate no advantage, from a methodological point of view, in using a large number (> 10) of starting solutions. We recommend to average results for a given QAP-heuristic over ten runs, but even five runs might suffice.

One should be careful to note that in this final experiment we have used the TA in a version which allows for rather intensive search and, thus, optimizes reliably. The picture might change when one employs a faster but less accurate optimizer. Though, results in Table 4 indicate that this would not be a very sensible strategy for QAP-optimization anyway.

5. Conclusions

We have presented a modified TA scheme that utilizes a flexible threshold function instead of a prespecified vector of threshold values. The initial threshold value is determined by a random sample of solution trials. Also, an iteration function was intoduced to allow for more iterations in later search phases when improvements are more difficult to achieve. Three different parameter settings for the threshold and iteration function were tested, resulting in a very fast optimizer, a high quality optimizer and a compromise between these two. The three TA strategies were tested on a set of QAPs varying in size and structure. Numerical experiments confirmed that our modified TA scheme is an efficient, easily implementable heuristic which may be adapted for speed, solution quality or any compromise between these objectives.

A second set of experiments demonstrated that, given a fixed amount of CPU-time, it is more useful to allow for a few runs of intensive TA-search instead of restarting over and over again from different initial solutions with a quick TA-version.

Finally, the numerical results indicated that it suffices to investigate algorithmic performance on the basis of ten random starting solutions. Using more initial points (i.e. runs) does not appear to change the relevant QAP-statistics (mean, std. dev.) of the solution method significantly, while it does, of course, increase CPU-requirements.

Acknowledgements. We wish to thank the anonymous referees for their helpful comments as well as Dr. Finn Kirstein, Uwe Maurer and Claudia Rott for their technical assistance. The first author gratefully acknowledges financial support by the Stiftung Volkswagenwerk, Germany.

References

1. Burkard RE, Rendl F (1984) A Thermodynamically Motivated Simulation Procedure for Combinatorial Optimization Problems. EJOR 17: 169–174
2. Burkard RE (1990) Locations with Spatial Interactions: The Quadratic Assignment Problem. In Mirchandani PB, Francis RL (eds) Discrete Location Theory. John Wiley & Sons, New York
3. Burkard RE, Karisch S, Rendl F (1991) QAPLIB – A Quadratic Assignment Problem Library. EJOR 55: 115–119
4. Dueck G, Scheuer T (1990) Threshold Accepting: A General Purpose Optimization Algorithm Appearing Superior to Simulated Annealing. J Comput Phys 90: 161–175
5. Elshafei AN (1977) Hospital Layout as a Quadratic Assignment Problem. Oper Res Q 28 (1ii): 167–179
6. Hadley SW, Rendl F, Wolkowicz H (1992) A New Lower Bound via Projection for the Quadratic Assignment Problem. Math Oper Res 17: 727–739

7. Heragu SS, Kusiak A (1991) Efficient Models for the Facility Layout Problem. EJOR 53: 1–13
8. Heragu SS (guest ed.) (1992) Facility Layout (Special Issue). EJOR 57 (2): 135–304
9. Heragu SS, Alfa AS (1992) Experimental Analysis of Simulated Annealing Based Algorithms for the Layout Problem. EJOR 57 (2): 190–202
10. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by Simulated Annealing. Science 220: 671–680
11. Koopmans TC, Beckmann MJ (1957) Assignment Problems and the Location of Economic Activities. Econometrica 25: 53–76
12. Krarup J, Pruzan PM (1978) Computer-Aided Layout Design. Math Program Stud 9: 75–94
13. Kusiak A, Heragu SS (1987) The Facility Layout Problem. EJOR 29: 229–251
14. Nugent EN, Vollmann TE, Ruml J (1968) An Experimental Comparison of Techniques for the Assignment of Facilities to Locations. Oper Res 16: 150–173
15. Paulli J. (1993) Information Utilization in Simulated Annealing and Tabu Search. Committee on Algorithms Bulletin 22: 28–34
16. Sahni S, Gonzales T (1976) P-complete Approximation Problem. ACM J 23: 556–565
17. Skorin-Kapov J (1990) Tabu Search Applied to the Quadratic Assignment Problem. ORSA J Comput 2 (1): 33–45
18. Steinberg L (1961) The Backboard Wiring Problem. SIAM Rev 3: 37–50
19. Taillard E (1991) Robust TABU Search for the Quadratic Assignment Problem. Parallel Comput 17: 443–455
20. Vollmann TE, Buffa ES (1966) The Facility Layout Problem in Perspective. Manag Sci 12 (10): B450–B468
21. Voß S (1994) Solving Quadratic Assignment Problems Using the Reverse Elimination Method. IN: Nash SG, Sofer A (eds) The impact of emerging technologies on computer science and operations research. Kluwer, Dordrecht

This article was processed by the author using the LaTeX style file *pljour2* from Springer-Verlag.