# A PATH RELINKING APPROACH FOR THE GENERALIZED ASSIGNMENT PROBLEM

**M. Yagiura** **T. Ibaraki**
Department of Applied Mathematics and Physics
Graduate School of Informatics, Kyoto University
Kyoto 606-8501, Japan
Email: {yagiura, ibaraki}@i.kyoto-u.ac.jp

**F. Glover**
Leeds School of Business
University of Colorado
Boulder, Colorado 80309-0419
Email: fred.glover@colorado.edu

## ABSTRACT

The generalized assignment problem is a classical combinatorial optimization problem known to be NP-hard. It can model a variety of real world applications in location, allocation, machine assignment, and supply chains. Researchers have studied the problem since the late 1960s, and computer codes for practical applications emerged in the early 1970s. We propose a new algorithm for this problem which proves to be more effective than previously existing methods. The algorithm features a path relinking approach, which is a mechanism for generating new solutions by combining two or more reference solutions. Computational comparisons on benchmark instances show that the method is not only effective in general, but is especially effective for the types D and E instances of the generalized assignment problem, which are known to be quite difficult.

## INTRODUCTION

We introduce an effective metaheuristic algorithm for the generalized assignment problem (GAP), which is one of the representative combinatorial optimization problems known to be NP-hard. This problem has many important applications, notably including scheduling, supply chain, location and vehicle routing problems. Consequently, the challenge of designing a good heuristic algorithm for GAP has significant practical as well as theoretical value.

Our algorithm features a path relinking approach (Glover, 1994, 1997) associated with adaptive memory programming (tabu search), which provides an "evolutionary" mechanism for generating new solutions by combining two or more reference solutions. The algorithm also features an ejection chain approach, likewise associated with tabu search, which is embedded in a neighborhood construction to create more complex and powerful moves. Lagrangian relaxation provides adjusted cost information to guide the neighborhood search to promising solutions. Moreover, we incorporate an automatic mechanism for adjusting search parameters, to maintain a balance between visits to feasible and infeasible regions.

Computational comparisons are conducted on benchmark GAP instances known as types C, D and E. These test problems are taken from the OR-Library, which is the primary repository for such problems, and are supplemented by additional test instances generated by ourselves. The results show that our GAP method is highly effective, especially for the types D and E instances, which have been established as the most difficult problem classes.

Our proposed algorithm is confirmed by extensive computational experiment to be efficient and robust, both in relation to parameter settings and variations in problem structure. The outcomes indicate that useful benefits result by combining path relinking and ejection chain strategies associated with adaptive memory methods, and making use of classical relaxation methodology. The resulting method yields a powerful and effective tool for practical applications.

## GENERALIZED ASSIGNMENT PROBLEM

Given $n$ jobs $J = \{1, 2, \ldots, n\}$ and $m$ agents $I = \{1, 2, \ldots, m\}$, we undertake to determine a minimum cost assignment subject to assigning each job to exactly one agent and satisfying a resource constraint for each agent. Assigning job $j$ to agent $i$ incurs a cost of $c_{ij}$ and consumes an amount $a_{ij}$ of resource, whereas the total amount of the resource available at agent $i$ is $b_i$. An assignment is a mapping $\sigma$: $J \to I$, where $\sigma(j) = i$ means that job $j$ is assigned to agent $i$. Then the *generalized assignment problem* (GAP) is formulated as follows:

$$\begin{aligned} &\text{minimize} \quad cost(\sigma) = \sum_{j \in J} c_{\sigma(j),j} \\ &\text{subject to} \quad \sum_{\substack{j \in J \\ \sigma(j)=i}} a_{ij} \le b_i, \quad \forall i \in I. \end{aligned} \tag{1}$$

GAP is known to be NP-hard, and the (supposedly) simpler problem of finding a feasible solution for GAP is also NP-hard, since the partition problem can be reduced to this problem with $m = 2$.

There are five types of benchmark instances called types A, B, C, D and E (Chu and Beasley, 1997; Laguna et al., 1995). Out of these, we use three types C, D and E, since other two are too easy to see differences among the tested algorithms. Types D and E are usually somewhat harder than type C, since $c_{ij}$ and $a_{ij}$ are inversely correlated. We tested 18 instances of types C, D and E with $n$ up to 200. Among them, types C and D instances were taken from OR-Library,[1] and type E instances were generated by ourselves, and are available at our WWW site.[2]

## ALGORITHM

Our algorithm, called PREC (path relinking & ejection chains), is an extension of local search. Local search starts from an initial solution $\sigma$ and repeatedly replaces $\sigma$ with a better solution in its *neighborhood* $N(\sigma)$ until no better solution is found in $N(\sigma)$. The resulting solution $\sigma$ is *locally optimal* in the sense that no better solution exists in its neighborhood. A shift neighborhood $N_{\text{shift}}$ is usually used in local search methods for GAP, where $N_{\text{shift}}(\sigma) = \{\sigma' \mid \sigma'$ is obtained from $\sigma$ by changing the assignment of one job$\}$. Our algorithm uses an ejection chain neighborhood, which consists of solutions obtainable by certain sequences of shift moves. Since the size of such a neighborhood can become exponential, we carefully limit its size by using the information from the Lagrangian relaxation of GAP. The resulting local search is called *EC probe*. For details of EC probe, see (Yagiura, Ibaraki and Glover, 1999).

When the search visits the infeasible region, we evaluate the solutions by an objective function penalized by infeasibility:

$$pcost(\sigma) = cost(\sigma) + \sum_{i \in I} \alpha_i p_i(\sigma), \tag{2}$$

where $p_i(\sigma) = \max\left\{0,\ \sum_{j \in J,\, \sigma(j)=i} a_{ij} - b_i\right\}$. The parameters $\alpha_i$ ($> 0$) are adaptively controlled during the search by using the algorithm in (Yagiura, Ibaraki and Glover, 1999).

The initial solutions for EC probes are generated by path relinking, which is a methodology to generate solutions from two or more solutions. Here we generate a sequence $\sigma_0, \sigma_1, \sigma_2, \ldots$ of solutions from two solutions $\sigma_A$ and $\sigma_B$ as follows. First we set $\sigma_0 := \sigma_A$. Then, for each $k = 1, 2, \ldots$, we define $\sigma_k$ to be the solution in $N_{\text{shift}}(\sigma_{k-1})$ with the best *pcost* among those whose distance to $\sigma_B$ is smaller than that from $\sigma_{k-1}$, where the distance between two solutions $\sigma$ and $\sigma'$ is defined to be $|\{j \in J \mid \sigma(j) \neq \sigma'(j)\}|$. Let $d$ be the distance between solutions $\sigma_A$ and $\sigma_B$. Then $\sigma_d = \sigma_B$ holds. We keep the best $\gamma$ (a parameter) solutions from $\{\sigma_1, \sigma_2, \ldots, \sigma_{d-1}\}$ in a heap, and use them as the initial solutions for EC probes.

[1] URL of OR-Library: http://mscmga.ms.ic.ac.uk/jeb/orlib/gapinfo.html

[2] URL of our WWW site: http://www-or.amp.i.kyoto-u.ac.jp/~yagiura/gap/

Table 1. COMPARISON OF PATH RELINKING AND UNIFORM CROSSOVER

| | | | path relinking | | | uniform crossover | | |
|---|---|---|---|---|---|---|---|---|
| type | $n$ | $m$ | min | avg. | max | min | avg. | max |
| C | 100 | 5 | 1931 | *1931.0 | 1931 | 1931 | *1931.0 | 1931 |
| C | 100 | 10 | 1402 | *1402.0 | 1402 | 1402 | *1402.0 | 1402 |
| C | 100 | 20 | 1243 | *1243.0 | 1243 | 1243 | *1243.0 | 1243 |
| C | 200 | 5 | 3456 | *3456.0 | 3456 | 3456 | *3456.0 | 3456 |
| C | 200 | 10 | 2806 | *2806.8 | 2807 | 2806 | *2806.8 | 2807 |
| C | 200 | 20 | 2391 | 2391.6 | 2392 | 2391 | *2391.0 | 2391 |
| D | 100 | 5 | 6353 | *6353.2 | 6354 | 6353 | 6354.0 | 6356 |
| D | 100 | 10 | 6348 | *6353.0 | 6358 | 6359 | 6363.4 | 6370 |
| D | 100 | 20 | 6210 | *6215.2 | 6220 | 6259 | 6269.0 | 6275 |
| D | 200 | 5 | 12744 | 12744.8 | 12746 | 12744 | *12744.6 | 12746 |
| D | 200 | 10 | 12436 | *12438.6 | 12441 | 12464 | 12468.0 | 12473 |
| D | 200 | 20 | 12266 | *12268.4 | 12271 | 12349 | 12356.4 | 12363 |
| E | 100 | 5 | 12681 | *12681.0 | 12681 | 12681 | 12681.2 | 12682 |
| E | 100 | 10 | 11577 | *11577.0 | 11577 | 11577 | *11577.0 | 11577 |
| E | 100 | 20 | 8436 | *8441.0 | 8444 | 8526 | 8547.4 | 8563 |
| E | 200 | 5 | 24930 | *24930.0 | 24930 | 24930 | 24930.2 | 24931 |
| E | 200 | 10 | 23307 | *23307.8 | 23308 | 23312 | 23313.0 | 23315 |
| E | 200 | 20 | 22384 | *22385.8 | 22388 | 22451 | 22493.0 | 22528 |

Whenever path relinking is applied, the two solutions $\sigma_A$ and $\sigma_B$ are randomly chosen from a reference set $R$ ($|R|$ is a parameter) of good solutions. Then, before applying the path relinking, a random shift is applied to $\sigma_B$ with probability 1/2 (no shift is applied with probability 1/2) to keep the diversity of the search. Initially $R$ is generated randomly. Once a feasible solution is found, one of the solutions in $R$ is the incumbent solution (i.e., the best feasible solution). Other solutions in $R$ are maintained as follows. Whenever an EC probe stops, the locally optimal solution $\sigma_{\text{lopt}}$ is exchanged with the worst solution $\sigma_{\text{worst}}$ in $R$ (excluding the incumbent solution), under the condition that $\sigma_{\text{lopt}}$ is not worse than $\sigma_{\text{worst}}$ and is different from all solutions in $R$.

## COMPUTATIONAL RESULTS

In this section, algorithm PREC is evaluated on the benchmark instances. All the algorithms were coded in C and run on a workstation Sun Ultra 2 Model 2300 (two UltraSPARC II 300MHz processors with 1 GB memory), where the computation was executed on a single processor. The parameters were set to $|R| = 20$ and $\gamma = 10$.

### Path Relinking and Uniform Crossover

We first compared the path relinking approach with the application of uniform crossover. Uniform crossover is one of the traditional methods to generate a new solution by combining two (or more) solutions, which is often used in genetic algorithms. The uniform crossover operation as normally defined must be amended to maintain the assignment structure of GAP solutions,

which we accomplish in a simple manner by making reference to the $\sigma$ mapping as follows. A new solution $\sigma_{new}$ is generated from two solutions $\sigma_A$ and $\sigma_B$ by randomly choosing $\sigma_{new}(j)$ from $\{\sigma_A(j), \sigma_B(j)\}$ with probability 1/2 for each $j \in J$ independently.

Table 1 shows the comparison of the two approaches. The two algorithms are exactly the same except for these solution generation mechanisms, i.e., we took full advantage of all of the ejection chain methodology and the Lagrangian relaxation component in the uniform crossover version. Each algorithm was executed five times for each instance, and the minimum, average and maximum costs are reported. The time limit for each run is 150 (resp., 300) seconds for instances with $n = 100$ (resp., 200). The mark '$*$' means that a better (or the same) average cost is attained. For the type C instances, which are quite easy for both methods, no significant differences emerge. However, for the more challenging type D and E instances, the path relinking approach clearly dominates the uniform crossover approach. In fact, for the type D and E instances, the worst result (as shown in the column "max") obtained by path relinking is often better than the best result (as shown in the column "min") obtained by uniform crossover (e.g., type D, $n = 100$, $m = 10$).

## Comparison with Other Heuristics

Algorithm PREC was compared with eight heuristic algorithms: (1) tabu search based on ejection chains by (Yagiura, Ibaraki and Glover, 1999) (denoted TSEC), (2) two algorithms of branching variable depth search by (Yagiura, Yamaguchi and Ibaraki, 1998) (denoted BVDS-l and BVDS-j), (3) variable depth search by (Yagiura, Yamaguchi and Ibaraki, 1999) (denoted VDS), (4) variable depth search by (Racer and Amini, 1994) (denoted RA), (5) tabu search by (Laguna et al., 1995) (denoted LKGG), (6) tabu search for the general purpose constraint satisfaction problem by (Nonobe and Ibaraki, 1998) (denoted NI), (7) a MAX-MIN ant system combined with local search and tabu search by (Lourenço and Serra, 1998) (denoted RLS). TSEC, BVDS-l, BVDS-j, VDS and RA were coded in C language by ourselves, while the codes of LKGG, NI and RLS were provided by the authors. The codes of LKGG and NI are written in C, and that of RLS is written in FORTRAN 77. The parameters for BVDS-l, BVDS-j and VDS are set to the values reported in (Yagiura, Yamaguchi and Ibaraki, 1998). RA does not include any parameter. The parameters for LKGG and NI are set to the default values. The RLS codes include various types of algorithms, which can be combined by choosing appropriate options. Here we chose the option ASH+LS+TS as recommended in (Lourenço and Serra, 1998), and other parameters were set to the default values. For comparison purposes, we also include in Table 2 the results of the genetic algorithm by (Chu and Beasley, 1997) (denoted CB), and the tabu search by (Díaz and Fernández, 2001) (denoted DF). The results of CB for type E instances are not available, and are denoted 'N.A.' in the table.

Table 2 shows the best costs obtained by these algorithms within 150 seconds for $n = 100$, and 300 seconds for $n = 200$, respectively, unless otherwise stated below the table. See (Yagiura, Ibaraki and Glover, 1999) for the computational time of RLS and CB. According to the estimate in (Díaz and Fernández, 2001), the total time of each run of algorithm DF is smaller than ours; however, the results in column DF are the best of 30 runs, and if this fact is taken into consideration, the total time of DF is larger than ours. In Table 2, we also show the lower bounds (denoted LB) obtained by solving the Lagrangian relaxation of GAP. In the table, each '$*$' mark indicates that the best cost is attained, and '—' means that no feasible solution was found. From the table, we can observe that the proposed PREC is very effective, especially for type D instances.

## Detailed Results of Our Algorithm

As a foundation for encouraging future research, we also show the detailed results of the proposed PREC method in Table 3 with a longer time limit for each run: 3000 seconds for $n = 100$, and 6000 seconds for $n = 200$, respectively. The table shows the minimum, average and maximum cost, the number of runs where the best solution is found, and the average computational time to find the best solution (over those runs in which the best cost is found) among five runs of algorithm PREC. In this table, we also show the best known solutions in (Yagiura, Ibaraki and Glover, 1999), where the values with '$\dagger$' are known to be optimal. The mark '$*$' (resp., '$**$') means that a tie (resp., better) solution is found. We can observe that three new best solutions were found for type D instances. For most of the instances whose optimal solutions are known, algorithm PREC found the optimal solutions in all of the five runs.

## CONCLUSION

The proposed path relinking approach (PREC) proves to be highly effective for the generalized assignment problem. Isolating the path relinking component of our algorithm and comparing it to the use of a uniform crossover component (a traditional mean for combining solutions often reported to be a method of choice) discloses that the outcomes from path relinking are significantly superior to those of uniform crossover. More extensive comparisons of the PREC algorithm, testing against eight other methods that are the leading heuristic approaches for GAP, confirm the high quality of the method's performance in general. PREC found better solutions than the other methods for all tested type D instances, and many of the type C and E instances. Tests of PREC with longer computational time are also performed, yielding new best solutions for three type D instances.

Table 2. THE BEST COSTS OBTAINED BY THE TESTED ALGORITHMS

| type | $n$ | $m$ | LB | PREC | TSEC | BVDS-l | BVDS-j | YYI | RA | LKGG | NI | RLS$^{\star}$ | CB$^{\star\star}$ | DF$^{\star\star\star}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C | 100 | 5 | 1930 | *1931 | *1931 | *1931 | *1931 | *1931 | 1938 | *1931 | *1931 | 1942 | *1931 | *1931 |
| C | 100 | 10 | 1400 | *1402 | *1402 | *1402 | 1403 | *1402 | 1405 | 1403 | 1403 | 1407 | 1403 | *1402 |
| C | 100 | 20 | 1242 | *1243 | *1243 | 1244 | 1244 | 1246 | 1250 | 1245 | 1245 | 1247 | 1244 | *1243 |
| C | 200 | 5 | 3455 | *3456 | *3456 | *3456 | 3457 | 3457 | 3469 | 3457 | 3465 | 3467 | 3458 | 3457 |
| C | 200 | 10 | 2804 | 2807 | *2806 | 2809 | 2808 | 2809 | 2835 | 2812 | 2817 | 2818 | 2814 | 2807 |
| C | 200 | 20 | 2391 | 2392 | 2392 | 2401 | 2400 | 2405 | 2419 | 2396 | 2407 | 2405 | 2397 | *2391 |
| D | 100 | 5 | 6350 | *6353 | 6357 | 6358 | 6362 | 6365 | — | 6386 | 6415 | 6476 | 6373 | 6357 |
| D | 100 | 10 | 6342 | *6348 | 6358 | 6367 | 6370 | 6380 | 6532 | 6406 | 6487 | 6469 | 6379 | 6355 |
| D | 100 | 20 | 6177 | *6210 | 6221 | 6275 | 6245 | 6284 | 6428 | 6297 | 6368 | 6358 | 6269 | 6220 |
| D | 200 | 5 | 12741 | *12745 | 12746 | 12755 | 12755 | 12778 | — | 12788 | 12973 | 12923 | 12796 | 12747 |
| D | 200 | 10 | 12426 | *12436 | 12446 | 12480 | 12473 | 12496$^{\dagger}$ | 12799 | 12537 | 12889 | 12746 | 12601 | 12457 |
| D | 200 | 20 | 12230 | *12267 | 12284 | 12440 | 12318 | 12335$^{\dagger}$ | 12665 | 12436 | 12793 | 12617 | 12452 | 12351 |
| E | 100 | 5 | 12673 | *12681 | 12682 | *12681 | 12682 | 12685 | 12917 | 12687$^{\dagger\dagger}$ | 12686$^{\ddagger}$ | 12836 | N.A. | *12681 |
| E | 100 | 10 | 11568 | *11577 | *11577 | 11585 | 11599 | 11585 | 12047 | 11641$^{\dagger\dagger}$ | 11590$^{\ddagger}$ | 11780 | N.A. | 11581 |
| E | 100 | 20 | 8431 | 8444 | *8443 | 8499 | 8484 | 8490 | 9004 | 8522$^{\dagger\dagger}$ | 8509$^{\ddagger}$ | 8717 | N.A. | 8460 |
| E | 200 | 5 | 24927 | *24930 | *24930 | 24942 | 24933 | 24948 | 25649 | 25147$^{\dagger\dagger}$ | 24958$^{\ddagger}$ | 25317 | N.A. | 24931 |
| E | 200 | 10 | 23302 | 23308 | *23307 | 23346 | 23348 | 23340 | 24717 | 23567$^{\dagger\dagger}$ | 23396$^{\ddagger}$ | 23620 | N.A. | 23318 |
| E | 200 | 20 | 22377 | *22384 | 22391 | 22475 | 22437 | 22452$^{\dagger}$ | 24117 | 22659$^{\dagger\dagger}$ | 22551$^{\ddagger}$ | 22779 | N.A. | 22422 |

†Results after 1,000 seconds on Sun Ultra 2 Model 2300
††Results after 20,000 seconds on Sun Ultra 2 Model 2300
‡Results after 5,000 seconds on Sun Ultra 2 Model 2300
⋆Computational time is reported in (Yagiura, Ibaraki and Glover, 1999)
⋆⋆Results in (Chu and Beasley, 1997)
⋆⋆⋆Results in (Díaz and Fernández, 2001)

Table 3. DETAILED RESULTS OF ALGORITHM PREC

| | | | | best | cost of 5 runs | | | #best | avg. time | time |
|---|---|---|---|---|---|---|---|---|---|---|
| type | $n$ | $m$ | LB | known | min | avg. | max | found | to best | limit |
| C | 100 | 5 | 1930 | †1931 | *1931 | 1931.0 | 1931 | 5/5 | 0.7 | 3000 |
| C | 100 | 10 | 1400 | †1402 | *1402 | 1402.0 | 1402 | 5/5 | 2.8 | 3000 |
| C | 100 | 20 | 1242 | †1243 | *1243 | 1243.0 | 1243 | 5/5 | 32.3 | 3000 |
| C | 200 | 5 | 3455 | †3456 | *3456 | 3456.0 | 3456 | 5/5 | 10.7 | 6000 |
| C | 200 | 10 | 2804 | †2806 | *2806 | 2806.2 | 2807 | 4/5 | 1070.7 | 6000 |
| C | 200 | 20 | 2391 | †2391 | *2391 | 2391.0 | 2391 | 5/5 | 1804.3 | 6000 |
| D | 100 | 5 | 6350 | †6353 | *6353 | 6353.0 | 6353 | 5/5 | 83.6 | 3000 |
| D | 100 | 10 | 6342 | 6349 | **6348 | 6352.0 | 6356 | 1/5 | 7.1 | 3000 |
| D | 100 | 20 | 6177 | 6196 | 6197 | 6204.0 | 6209 | 1/5 | 2340.2 | 3000 |
| D | 200 | 5 | 12741 | 12743 | *12743 | 12743.4 | 12744 | 3/5 | 3565.1 | 6000 |
| D | 200 | 10 | 12426 | 12436 | **12433 | 12436.0 | 12438 | 1/5 | 480.3 | 6000 |
| D | 200 | 20 | 12230 | 12264 | **12244 | 12251.4 | 12258 | 1/5 | 5913.0 | 6000 |
| E | 100 | 5 | 12673 | †12681 | *12681 | 12681.0 | 12681 | 5/5 | 16.7 | 3000 |
| E | 100 | 10 | 11568 | †11577 | *11577 | 11577.0 | 11577 | 5/5 | 11.3 | 3000 |
| E | 100 | 20 | 8431 | †8436 | *8436 | 8437.6 | 8440 | 3/5 | 280.8 | 3000 |
| E | 200 | 5 | 24927 | †24930 | *24930 | 24930.0 | 24930 | 5/5 | 11.1 | 6000 |
| E | 200 | 10 | 23302 | †23307 | *23307 | 23307.0 | 23307 | 5/5 | 2034.9 | 6000 |
| E | 200 | 20 | 22377 | †22379 | *22379 | 22380.2 | 22385 | 4/5 | 1295.4 | 6000 |

†known to be optimal values

## REFERENCES


P.C. Chu and J.E. Beasley (1997) “A genetic algorithm for the generalized assignment problem,” *Computers Oper. Res.*, Vol. 24, pp. 17–23.

J.A. Díaz and E. Fernández (2001) “A tabu search heuristic for the generalized assignment problem,” *European J. Oper. Res.*, Vol. 132, pp. 22–38.

F. Glover (1994) “Genetic algorithms and scatter search: unsuspected potentials,” *Statistics and Computing*, Vol. 4, pp. 131–140.

F. Glover (1997) “A template for scatter search and path relinking,” *LNCS*, 1363, Springer, pp. 13–54.

M. Laguna, J.P. Kelly, J.L. González-Velarde and F. Glover (1995) “Tabu search for the multilevel generalized assignment problem,” *European J. Oper. Res.*, Vol. 82, pp. 176–189.

H.R. Lourenço and D. Serra (1998) “Adaptive approach heuristics for the generalized assignment problem,” Technical report available at http://www.econ.upf.es/deehome/what/wpapers/listwork.html.

K. Nonobe and T. Ibaraki (1998) “A tabu search approach to the CSP (constraint satisfaction problem) as a general problem solver,” *European J. Oper. Res.*, Vol. 106, pp. 599–623.

M. Racer and M.M. Amini (1994) “A robust heuristic for the generalized assignment problem,” *Ann. Oper. Res.*, Vol. 50, pp. 487–503.

M. Yagiura, T. Ibaraki and F. Glover (1999) “An ejection chain approach for the generalized assignment problem,” Technical Report #99013, Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University.

M. Yagiura, T. Yamaguchi and T. Ibaraki (1998) “A variable depth search algorithm with branching search for the generalized assignment problem,” *Optimization Methods and Software*, Vol. 10, pp. 419–441.

M. Yagiura, T. Yamaguchi and T. Ibaraki (1999) “A variable depth search algorithm for the generalized assignment problem,” in: S. Voß, S. Martello, I.H. Osman and C. Roucairol, eds., *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, Kluwer, pp. 459–471.