G9.10 Quadratic assignment

Volker Nissen

Abstract

The quadratic assignment problem (QAP) is of practical relevance in different fields of application, such as hospital layout planning, machine scheduling, and component placement on printed circuit cards. We discuss the QAP as a base model for facility layout problems and present an evolutionary solution technique derived from evolution strategy (ES), one of the mainstream forms of evolutionary algorithms (EAs). It is empirically evaluated on a test suite of QAPs. The implemented ES for combinatorial problems (CES) is deliberately not hybridized with other solution techniques. This gives an idea of the potential of ES on this standard problem in the light of recent debates about the competitiveness of EAs for solving combinatorial problems. CES is a good heuristic for solving QAPs and does not require tuning of strategy parameters on individual problem instances. Even though CES is superior over the classical 2-Opt procedure and an evolutionary approach proposed by Tate and Smith (1995), it is not fully competitive with Threshold Accepting, one of the most efficient heuristics currently available for QAPs. Implications of these results are discussed.

G9.10.1 Facility layout and quadratic assignment

Locating facilities with material flow between them, such as machines in a factory hall, is a difficult layout problem that is frequently modeled as a quadratic assignment problem (QAP) (Kusiak and Heragu 1987). The QAP is one of the most difficult combinatorial optimization problems and can be formalized as follows (Burkard 1990): given a set $N = \{1, 2, \ldots, n\}$ and real numbers $c_{ik}, a_{ik}, b_{lk}$ for $i, k = 1, 2, \ldots, n$, find a permutation $\phi$ of the set $N$ which minimizes

$$Z = \sum_{i=1}^{n} c_{i\phi(i)} + \sum_{i=1}^{n} \sum_{k=1}^{n} a_{ik} b_{\phi(i), \phi(k)}$$

(G9.10.1)

where $n$ is the total number of facilities and locations, $c_{ij}$ is the fixed cost of locating facility $j$ at location $i$, $b_{jl}$ is the flow of material from facility $j$ to facility $\lambda$, and $a_{ik}$ is the cost of transferring a material unit from location $i$ to location $k$.

The linear part in (G9.10.1) may be considered as installation costs while the quadratic part accounts for interaction (material traffic) between facilities. The QAP is characterized by a high degree of interaction between solution elements (assignments). Even swapping the assignment of two facilities might affect the quality of virtually all other assignments, depending on the flow matrix. As a generalization of the traveling salesman problem (TSP), the QAP is NP-hard, and only moderately sized problem instances ($n \simeq 18$) can be solved to optimality with exact algorithms within reasonable time limits. One therefore concentrates on developing heuristics for the QAP. Extensive reviews of the QAP and associated solution techniques can be found in the articles by Kusiak and Heragu (1987) and Burkard (1990).

Vollmann and Buffa (1966) introduced the concept of flow dominance, measuring the variation of values in the flow matrix. It is given by

$$100 \text{std dev.} / \text{mean}$$
of the matrix elements. Simply stated, high flow dominance indicates that a few facilities with high interaction tend to dominate the problem. Burkard and Fincke (1983) were the first to prove the asymptotic behavior of large randomly generated QAPs. This means that the relative difference between the worst and the optimal solution becomes arbitrarily small with a probability tending to unity as the problem size tends to infinity. Thus, instead of focusing only on the large random problems frequently cited in the literature, it is more appropriate to use a suite of test problems varying in size and structure, where flow dominance is one sensible measure to characterize the structure of a QAP. Here, a set of seven QAPs varying in size between \( n = 15 \) and \( n = 64 \) with different structure (flow dominance) has been employed. The problems were taken from the QAP library collected by Burkard et al (1991).

A number of authors have previously developed evolutionary approaches to solve QAPs (see overviews by Alander (1995) and Nissen (1995)). When high-quality results were achieved, the EA frequently had been hybridized with other well-known problem solving techniques such as simulated annealing or tabu search. It is, therefore, difficult to assess the potential of the evolutionary part of these heuristics for QAPs.

In this case study, a variant of evolution strategy (ES) is proposed to solve QAPs. Since ES was originally not invented for combinatorial optimization, the methodology was adapted to suit the needs of this application while staying in the evolutionary framework. However, solving the QAP can only be considered a first step to approaching real-world facility layout problems that frequently involve additional complex constraints. For instance, locations may be of unequal size, permitting only a subset of machines to be located at a certain position. Fixed costs for material transport can occur. Safety or technical considerations may yield certain assignments invalid. Accurate material flow data may not be available. Locations might not be determined in advance. In a multicriterion decision situation, aspects such as safety or flexibility of a layout as well as environmental objectives might be additional goals. However, only a few authors (e.g. Tam 1992, Kouvelis et al 1992, Smith and Tate 1993, Krause and Nissen 1995) have extended the QAP to include some such practically relevant considerations.

### G9.10.2 Design and implementation of the evolution strategy

Our ES variant for combinatorial problems, termed CES, was first presented by Nissen (1994b). It uses a straightforward permutation coding (figure G9.10.1). A simple population concept, basically a \((1, \lambda)\) ES, is employed. This means, in each generation \( \lambda \) offspring are generated from one parent solution. First, one produces \( \lambda \) copies of the parent. Then, each of these copies is mutated by (possibly repeated) pairwise exchange of randomly determined positions of facilities (assignments) on the given solution, thereby generating a population of offspring. Note that the mutation operator from standard ES, which is based on normally distributed random variables, is not adequate for such a permutation coding since it would yield invalid results. Crossover of solutions is not employed. \( \lambda = 50 \) for the smaller instances NUG15, NUG20 and ELS19. For the other problems \( \lambda = 100 \). The parent is eliminated after each generation.

![Figure G9.10.1](https://example.com/figure.png)

**Figure G9.10.1.** Representation of a solution for \( n = 7 \). A similar figure appeared in Nissen (1994b) (copyright 1994 IEEE).

The number of pairwise exchanges during mutation is restricted to be randomly either one or two. It can occasionally be zero, however, should the algorithm by chance choose the same position for a swap twice. Too many exchanges would generally deteriorate the objective function value of a given QAP solution due to the massive interactions (traffic) between the facilities.

The best offspring becomes the new parent. If the parent’s objective function value represents no improvement over the former parent, a counter is increased. The counter is reset to zero whenever a CES generation is successful, that is, an improvement is achieved. After a certain number of consecutive
unsuccessful generations $g_u$, a procedure called destabilization is executed. It was found empirically that $(n/10 + 2)$, where the result is rounded, is a good value for $g_u$. This parameter value was used in all experiments but no claim is made as to its optimality.

Destabilization is essentially a more intensive form of mutation. During this phase, the counter is set to zero and $\lambda$ offspring are created with increased mutation intensity. The number of swaps now randomly lies in the interval $[3, \ldots, 8]$. Thereby, individuals which differ more strongly from previous solutions are generated and the search shifts to a new area in the solution space. This helps to escape from local optima and counters the strong selection pressure in CES. Again, the best offspring is determined to become the new parent. Procedure destabilization is then terminated and the search continues as before until $t_{\text{max}}$, the maximum number of generations.

CES starts from a randomly generated initial solution. The best solution ever found by the heuristic is stored separately, and it is continuously updated during the search. This is the final result of the heuristic.

An overview of CES is given in the following pseudocode. The parameters were empirically determined and kept constant over all QAP experiments with the exception of the loop variable $i$ that runs from 1 to 50 for the smaller problems NUG15, NUG20 and ELS19.

\begin{verbatim}
Input: $t_{\text{max}}$
Output: $x^*$, the best solution ever found.

1 $t \leftarrow 0$;
2 create initial solution $x_0$ randomly;
3 $x^* \leftarrow x_0$;
4 fail_count $\leftarrow 0$;
5 for $t \leftarrow 1$ to $t_{\text{max}}$ do
6  $\tilde{f} \leftarrow f(x_0)$;
7  for $i \leftarrow 1$ to 100 do
8    $x_i \leftarrow x_0$; \{copy parent\}
9    sample no_swaps $\sim U(1, 2)$;
10   mutate $x_i$ by swapping assignments according to no_swaps;
11      \{details see main text\}
12   evaluate $f(x_i)$;
13  od
14 $x_0 \leftarrow x_j$ where $f(x_j) = \min \{f(x_i) \mid i = 1, \ldots, 100\}$; \{selection\}
15 if ($f(x_0) \geq \tilde{f}$)
16    then increment fail_count;
17    else fail_count$\leftarrow 0$;
18      check if update of $x^*$ necessary;
19  fi
20 if (fail_count = round($n/10 + 2$))
21  then destabilization;
22 od
23 output $x^*$;
\end{verbatim}

\textit{procedure destabilization} ($x_0$)

1 for $i \leftarrow 1$ to 100 do
2   $x_i \leftarrow x_0$; \{copy parent\}
3   sample no_swaps $\sim U(3, 8)$;
4   mutate $x_i$ by swapping assignments according to no_swaps;
5      \{details see main text\}
6   evaluate $f(x_i)$;
7 od
8 $x_0 \leftarrow x_j$ where $f(x_j) = \min \{f(x_i) \mid i = 1, \ldots, 100\}$; \{selection\}
9 check if update of $x^*$ necessary;
10 fail_count$\leftarrow 0$;
11 return ($x_0$);
G9.10.3 Empirical results

G9.10.3.1 Performance of CES on the test suite

CES was run on seven test problems originally published by Nugent et al (1968) (NUG15, NUG20, NUG30), Steinberg (1961) (STE36a, STE36c), Elshafei (1977) (ELS19), and Skorin-Kapov (1990) (SKO64) with numbers of locations and facilities n (including dummy facilities) varying between 15 and 64. NUG15, NUG20, NUG30, and SKO64 are randomly generated problems with low flow dominance. The other three appear to be practical applications with high (STE36a, STE36c) and very high (ELS19) flow dominance values. The search space size (number of solution alternatives) varied from roughly $1.31 \times 10^{12}$ for NUG15 to $1.27 \times 10^{89}$ for SKO64. CES was implemented in Pascal on a workstation IBM RS 6000/320. Ten runs were performed in each experiment. It should be noted that in the work of Nissen and Paul (1995) we alternatively used five, ten and 30 initial solutions (= different runs) for evaluating another QAP heuristic. The performance measures, mean and standard deviation (std dev.) of the best objective function values from different runs, were apparently unaffected by this choice of the number of runs. Results for CES are given in table G9.10.1. Data for generation 0 refer to initial solutions.

A typical convergence chart for CES appears in figure G9.10.2. Generally, the optimal solution was approached in an asymptotic manner. Improvements were a little less continuous on ELS19, the problem with the highest flow dominance value. When flow dominance is very high, heuristics based on pairwise exchanges of assignments have difficulties in overcoming the pronounced local suboptima. Crossover can be advantageous in such a case, as was found in an empirical investigation on QAPs involving genetic algorithms (Nissen 1994a).

However, due to the chosen search operator based on pair exchanges CES allows for an efficient form of solution evaluation that could not be applied when crossover was used (see Nissen 1994a for details).

![Figure G9.10.2. A typical convergence chart for CES: the best and worst of ten runs on NUG30. A similar figure appeared in Nissen (1994b) (copyright 1994 IEEE).](image)

CES identified good solutions on all seven problem instances. Destabilization proved to be a useful heuristic element. On larger QAPs, there was a tendency for fewer destabilization phases. One reason is the way the allowed number of consecutive unsuccessful CES generations before destabilization is computed. The larger $n$, the higher this maximum value. Besides, with increasing problem dimension, it becomes easier to leave a local optimum. However, because the size of the search space ($n!$) rises drastically, more time is needed to identify high-quality solutions as compared to smaller problem instances.

CES is quite a useful heuristic for solving QAPs. It has acceptable CPU requirements, is easily implementable, and yields good results without problem-specific parameter tuning on QAPs of very different sizes and structures.
Table G9.10.1. CES results, starting from random initial solutions. Mean OFV is the mean objective function value of the best solution found up to this generation, averaged over ten runs on an IBM RS 6000/320. AFE is the average total number of function evaluations per run up to this generation.

<table>
<thead>
<tr>
<th>Test problem</th>
<th>Best known solution</th>
<th>Generation</th>
<th>Mean OFV</th>
<th>AFE</th>
<th>CPU (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUG15</td>
<td>0</td>
<td>1564</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1150</td>
<td>1162</td>
<td>10,811</td>
<td>1.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1,000</td>
<td>1153</td>
<td>54,446</td>
<td>7.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2,000</td>
<td>1151</td>
<td>108,976</td>
<td>14.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6,000</td>
<td>1150</td>
<td>327,086</td>
<td>44.2</td>
<td></td>
</tr>
<tr>
<td>NUG20</td>
<td>0</td>
<td>3,442</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2,570</td>
<td>2,626</td>
<td>10,641</td>
<td>2.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1,000</td>
<td>2,592</td>
<td>53,276</td>
<td>10.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2,000</td>
<td>2,587</td>
<td>106,626</td>
<td>21.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6,000</td>
<td>2,574</td>
<td>320,096</td>
<td>62.9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>20,000</td>
<td>2,570</td>
<td>1,067,451</td>
<td>209.6</td>
<td></td>
</tr>
<tr>
<td>NUG30</td>
<td>0</td>
<td>8,127</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6,124</td>
<td>6,310</td>
<td>10,291</td>
<td>3.6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1,000</td>
<td>6,202</td>
<td>52,281</td>
<td>18.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>6,166</td>
<td>104,701</td>
<td>37.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3,000</td>
<td>6,145</td>
<td>314,691</td>
<td>111.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10,000</td>
<td>6,135</td>
<td>1,050,181</td>
<td>372.4</td>
<td></td>
</tr>
<tr>
<td>SKO64</td>
<td>0</td>
<td>20,000</td>
<td>1,067,451</td>
<td>209.6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4,8498</td>
<td>20,000</td>
<td>1,067,451</td>
<td>209.6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>50,196</td>
<td>10,001</td>
<td>19.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>49,565</td>
<td>50,271</td>
<td>95.9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1,000</td>
<td>49,380</td>
<td>100,781</td>
<td>192.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3,000</td>
<td>49,044</td>
<td>302,591</td>
<td>583.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10,000</td>
<td>48,906</td>
<td>1,009,211</td>
<td>1927.5</td>
<td></td>
</tr>
<tr>
<td>ELS19</td>
<td>0</td>
<td>3,442</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>17,212548</td>
<td>20,000</td>
<td>10,486</td>
<td>1.9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>18,128,394</td>
<td>52,916</td>
<td>9.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1,000</td>
<td>18,128,394</td>
<td>105,851</td>
<td>19.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2,000</td>
<td>18,128,394</td>
<td>317,621</td>
<td>58.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6,000</td>
<td>17,517,830</td>
<td>317,621</td>
<td>58.2</td>
<td></td>
</tr>
<tr>
<td>STE36a</td>
<td>0</td>
<td>22,755</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>9,526</td>
<td>10,650</td>
<td>10,171</td>
<td>4.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>10,103</td>
<td>51,461</td>
<td>24.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>9,979</td>
<td>102,981</td>
<td>48.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1,000</td>
<td>9,812</td>
<td>309,561</td>
<td>146.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3,000</td>
<td>8,403</td>
<td>309,141</td>
<td>147.6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10,000</td>
<td>8,338</td>
<td>1,031,451</td>
<td>492.3</td>
<td></td>
</tr>
<tr>
<td>STE36c</td>
<td>0</td>
<td>18,890</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8,239.1</td>
<td>8,923</td>
<td>10,131</td>
<td>4.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>8,713</td>
<td>51,301</td>
<td>24.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>8,582</td>
<td>102,901</td>
<td>49.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1,000</td>
<td>8,403</td>
<td>309,141</td>
<td>147.6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3,000</td>
<td>8,338</td>
<td>1,031,451</td>
<td>492.3</td>
<td></td>
</tr>
</tbody>
</table>

G9.10.3.2 Comparison of CES with other approaches

The well known traditional combinatorial heuristic 2-Opt served as a first benchmark to compare the quality of solutions generated by CES with some other QAP heuristics (tables G9.10.2 and G9.10.3). 2-Opt is a simple local search heuristic that sequentially considers pairwise exchange between the positions of facilities. The swap is made whenever this results in a lower objective function value and the search starts again from the new solution. This procedure continues until no exchange of assignments in the current solution results in a further improvement. 2-Opt was implemented in Pascal on an IBM RS 6000/320, as was CES. The initial solutions of 2-Opt in table G9.10.2 were identical to those of CES.

CES on average quickly produced far better solutions than 2-Opt. Even after only 100 generations CES often generated better mean values. Moreover, it converged to better solutions with greater reliability as shown by the small std dev. in table G9.10.3.

Tate and Smith (1995) also proposed an evolutionary heuristic for solving QAPs. While they termed
Table G9.10.2. 2-Opt results with initial solutions identical to CES. Mean OFV is the mean objective function value of the best solution found, averaged over ten runs on an IBM RS 6000/320. AFE is the average total number of function evaluations per run.

<table>
<thead>
<tr>
<th>Test problem</th>
<th>Mean OFV</th>
<th>AFE</th>
<th>Avg. CPU (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUG15</td>
<td>1194</td>
<td>746</td>
<td>0.1</td>
</tr>
<tr>
<td>NUG20</td>
<td>2671</td>
<td>2002</td>
<td>0.2</td>
</tr>
<tr>
<td>NUG30</td>
<td>6322</td>
<td>10322</td>
<td>2.2</td>
</tr>
<tr>
<td>SKO64</td>
<td>49524</td>
<td>188650</td>
<td>241.6</td>
</tr>
<tr>
<td>ELS19</td>
<td>21798726</td>
<td>3594</td>
<td>0.4</td>
</tr>
<tr>
<td>STE36a</td>
<td>10447</td>
<td>17839</td>
<td>5.2</td>
</tr>
<tr>
<td>STE36c</td>
<td>8903</td>
<td>20188</td>
<td>5.9</td>
</tr>
</tbody>
</table>

Table G9.10.3. A comparison of CES and 2-Opt on NUG30 with respect to solution quality at approximately identical CPU requirements. All initial solutions were generated randomly. A similar table appeared in Nissen (1994b) (copyright 1994 IEEE).

<table>
<thead>
<tr>
<th></th>
<th>10 × CES (10 000 generations)</th>
<th>1700 × 2-Opt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best</td>
<td>Worst</td>
<td>Mean</td>
</tr>
<tr>
<td>6124</td>
<td>6150</td>
<td>6135</td>
</tr>
</tbody>
</table>

Table G9.10.4. Results of the evolutionary heuristic (25% crossover, 75% mutation) presented by Tate and Smith (1995) for test problems also used here. Mean OFV is the mean objective function value of the best solution found, averaged over ten runs. (Original values are doubled to account for symmetrical flow.)

<table>
<thead>
<tr>
<th>Test problem</th>
<th>Mean OFV</th>
<th>Funct. eval. per run</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUG15</td>
<td>1170</td>
<td>≈ 200 000</td>
</tr>
<tr>
<td>NUG20</td>
<td>2643</td>
<td>200 000</td>
</tr>
<tr>
<td>NUG30</td>
<td>6305</td>
<td>200 000</td>
</tr>
</tbody>
</table>

If a genetic algorithm, it is quite close to an evolution strategy, using some sort of \((\mu + \lambda)\)-selection and focusing on mutation rather than crossover. Some results with this approach for test problems also used here are reported in table G9.10.4. Since different hardware and software was used in the implementation of CES and the heuristic by Tate and Smith, we do not mention CPU times but function evaluations to account for computational effort. CPU requirements are primarily influenced by the calculation of objective function values in this application.

Let the efficiency of a search procedure be defined as the ratio of solution quality and required search effort. CES clearly showed a better performance than the other heuristic in terms of efficiency, averaged over ten runs. Moreover, Tate and Smith estimated the increase in computational effort per solution generated to be quadratic with the number of sites for their heuristic, which is higher than for CES and other approaches based on pairwise exchange of assignments, such as tabu search (see e.g. Fleurent and Ferland 1994).

Threshold Accepting (TA) is a local search technique and a much stronger competitor than 2-Opt or the evolutionary heuristic by Tate and Smith. TA is a simplification of the well known simulated annealing procedure that was initially proposed by Dueck and Scheuer (1990). Starting from an initial solution each TA step consists of a slight change of the old solution into a new one. Then, the qualities of the two solutions are compared with respect to the given objective function. 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 old solution as a basis for the next TA step. The threshold \(T\) will be relatively large at 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 increasing 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. Nissen and Paul (1995) modified the basic TA heuristic in several ways and applied it to the QAP. In this implementation, new solutions were obtained from a given configuration by a simple random pair exchange of assignments. The modified TA scheme appears to be one of the most efficient heuristics currently available for the QAP.

Table G9.10.5. Comparison of CES and TA for various run lengths. Data for TA was partly taken from Nissen and Paul (1995). Since hardware was different not CPU time but function evaluations are reported to compare efficiency. Results are based on 10 runs each with random initial solutions. AFE = average total number of function evaluations per run. For CES and TA, the total number of evaluations varied only very slightly between different runs on the same test problem.

<table>
<thead>
<tr>
<th>Test problem</th>
<th>CES</th>
<th>TA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Best</td>
<td>Mean</td>
</tr>
<tr>
<td>NUG20</td>
<td>2,580</td>
<td>2,626</td>
</tr>
<tr>
<td>2570</td>
<td>2,570</td>
<td>2,587</td>
</tr>
<tr>
<td>NUG</td>
<td>6,220</td>
<td>6,310</td>
</tr>
<tr>
<td>6124</td>
<td>6,150</td>
<td>6,202</td>
</tr>
<tr>
<td>SKO64</td>
<td>6,128</td>
<td>6,145</td>
</tr>
<tr>
<td>48498</td>
<td>49,720</td>
<td>50,196</td>
</tr>
<tr>
<td>48,872</td>
<td>49,044</td>
<td>302,591</td>
</tr>
<tr>
<td>ELS19</td>
<td>17,212,548</td>
<td>17,212,548</td>
</tr>
<tr>
<td>17,212,548</td>
<td>19,218,337</td>
<td>10,486</td>
</tr>
<tr>
<td>17,212,548</td>
<td>18,128,394</td>
<td>52,916</td>
</tr>
<tr>
<td>17,212,548</td>
<td>17,517,830</td>
<td>317,621</td>
</tr>
<tr>
<td>STE36a</td>
<td>10,218</td>
<td>10,650</td>
</tr>
<tr>
<td>9526</td>
<td>9,798</td>
<td>9,979</td>
</tr>
<tr>
<td>9,564</td>
<td>9,701</td>
<td>1,032,511</td>
</tr>
</tbody>
</table>

In table G9.11.5, CES and TA are compared in terms of their efficiency. Again, the comparison of computational requirements is based on function evaluations and not CPU time for the same reasons as before. The rise in computational effort per solution generated is roughly comparable for CES and TA as the problem size increases.

TA was superior over CES in all cases but ELS19 with its particularly high flow dominance and comparatively low dimensionality. This instance is difficult for any heuristic that only relies on a simple pair exchange of assignments. One could probably improve the performance of TA on this problem by introducing a destabilization (as in CES) or by transferring the idea of crossover from genetic algorithms. Even though CES is a good heuristic, one must conclude that it is not fully competitive with the best solution techniques currently available for the QAP.

G9.10.4 Conclusions

The variant of evolution strategy for combinatorial problems (CES) proposed here compared favorably with the classic 2-Opt procedure and results of an evolutionary heuristic by Tate and Smith on the quadratic assignment problem. The destabilization operator in our ES implementation is useful in overcoming local optima when selection pressure is high, as in CES. It is also successful on problem instances with high flow dominance that prove difficult for heuristics purely based on pairwise exchange of assignments. CES is a good heuristic for the QAP that, moreover, requires no tuning of strategy parameters on individual problem instances, making it user friendly. A more detailed description of the experimental setup, further experiments with genetic algorithms and evolutionary programming, and results of sensitivity analysis are given by Nissen (1994a).

CES could not fully compete with Threshold Accepting, however, one of the most efficient heuristics currently available for this application. This might fuel the debate on the potential of EAs in combinatorial optimization. However, one should recall that in our experiments hybridizing with other solution techniques was deliberately avoided. Fleurent and Ferland (1994), for instance, developed such hybrids for the QAP.
They successfully combined genetic algorithms with tabu search to produce competitive optimization techniques that were able to improve upon the best known solutions for a number of large random QAP test problems. However, the CPU requirements of their hybrids to achieve these improvements were very high (up to a few days on a SPARC 10 for some problem instances). One gets the feeling that, at least for the QAP, the competitiveness of the evolutionary approach to other modern heuristics actually depends on whether CPU requirements are of importance or parallel hardware is available, respectively. In facility layout, one can usually ignore CPU time so that carefully hybridized EAs can be competitive solution techniques.

References

Elshafei A N 1977 Hospital layout as a quadratic assignment problem Operat. Res. Q. 28 167–79
Nissen V 1994a Evolutionäre Algorithmen Darstellung, Beispiele, betriebswirtschaftliche Anwendungsmöglichkeiten. (Wiesbaden: DUV)
—1994b Solving the quadratic assignment problem with clues from nature IEEE Trans. Neural Networks: Special Issue on Evolutionary Programming NN-5 66–72
Nissen V and Paul H 1995 A modification of threshold accepting and its application to the quadratic assignment problem OR Spektrum 17 205–10
Nugent E N, Vollmann T E and Rumli J 1968 An experimental comparison of techniques for the assignment of facilities to locations Operat. Res. 16 150–73
Skorin-Kapov J 1990 Tabu search applied to the quadratic assignment problem ORSA J. Comput. 2 33–45
Smith A E and Tate D M 1993 Genetic optimization using a penalty function Proc. 5th Int. Conf. on Genetic Algorithms (Urbana-Champaign, July 1993 ed S Forrest (San Mateo, CA: Morgan-Kaufmann) pp 499–503
Steinberg L 1961 The backboard wiring problem SIAM Rev. 3 37–50