INFORMATICA, 2000, Vol. 11, No. 2, 145–162

© 2000 *Institute of Mathematics and Informatics, Vilnius*

An Intensive Search Algorithm for the Quadratic Assignment Problem

Alfonsas MISEVIČIUS
Kaunas University of Technology
Studentų St. 50-416D, LT-3031 Kaunas, Lithuania
e-mail: misevi@soften.ktu.lt

Received: March 2000

Abstract. Many heuristics, such as simulated annealing, genetic algorithms, greedy randomized adaptive search procedures are stochastic. In this paper, we propose a deterministic heuristic algorithm, which is applied to the quadratic assignment problem. We refer this algorithm to as intensive search algorithm (or briefly intensive search). We tested our algorithm on the various instances from the library of the QAP instances – QAPLIB. The results obtained from the experiments show that the proposed algorithm appears superior, in many cases, to the well-known algorithm – simulated annealing.

Key words: heuristics, intensive search, quadratic assignment problem.

1. Introduction

The quadratic assignment problem (QAP) is formulated as follows. Given matrices $F = (f_{ij})_{n\times n}$ and $D = (d_{kl})_{n\times n}$ and the set Π of all permutations of $1, 2, \ldots, n$. Find a permutation $p \in \Pi$ that minimizes

$$z(p) = \sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij} d_{p(i)p(j)}. \tag{1}$$

One of the applications of the quadratic assignment problem is computer aided design of electronic devices, namely, placement of the components (Steinberg, 1961). In this context, n denotes the number of components (blocks), which are to be placed into positions (slots) on a board (chip). Each placement corresponds to a certain permutation $(p(1), p(2), \ldots, p(n))$, where $p(i)$ denotes the index of the position that component i is placed into $(i = 1, 2, \ldots, n)$. The matrix $F = (f_{ij})$ can be interpreted as a matrix of connections between components, where f_{ij} is the sum of the nets connecting component i and component j. The matrix $D = (d_{kl})$ is the distance matrix, where d_{kl} represents the distance from position k to position l. Thus, z can be treated as a total estimated wire length (sum of the half-perimeters of the nets) obtained when n components are placed into n positions. The goal is to find a placement of all components into all positions, such

that the total estimated wire length is minimized. Further, we will consider the quadratic assignment problem to model the placement of components.

The quadratic assignment problem is one of the most complex combinatorial optimization problems. It has been proved that the quadratic assignment problem is NP-hard. There exists no polynomial-time algorithm to find an exact solution. Problems of size $n > 20$ are not, to this date, practically solvable to optimality (Mautor and Roucairol, 1994; Clausen and Perregaard, 1997). Therefore, heuristic algorithms have been widely used for the quadratic assignment problem, first of all, simulated annealing (Burkard and Rendl, 1984; Wilhelm and Ward, 1987; Thonemann and Boelte, 1994), tabu search (Skorin-Kapov, 1990; Taillard, 1991), genetic algorithms (Fleurent and Ferland, 1994), as well as the algorithms based on the greedy randomized adaptive search (Li *et al*., 1993), and other algorithms (Armour and Buffa, 1963; Bazaraa and Sherali, 1980; Sherali and Rajgopal, 1986; Murthy *et al*., 1992).

In this paper, we introduce a new heuristic algorithm entitled as an intensive search (IS) algorithm. We describe the details of this algorithm in Section 2. The results of the computational experiments, as well as a comparison of the proposed algorithm with simulated annealing algorithm, are presented in Section 3.

2. An Intensive Search Algorithm for the Quadratic Assignment Problem

Our heuristic algorithm works in combination with an algorithm for the initial solution generation; i.e., we use two-phase strategy for the quadratic assignment problem:

- INITIAL SOLUTION CONSTRUCTION
- SOLUTION IMPROVEMENT.

At the first phase, an initial solution (placement of the components) is generated using one of the constructive algorithms (see below). Then at the second phase the initial solution is improved by the intensive search algorithm.

2.1. *Construction of the Initial Solution*

Many constructive algorithms use a consecutive scheme of handling of the components (e.g., Gilmore, 1962; Hanan and Kurtzberg, 1972; Murthy and Pardalos, 1990). The components are considered successively: at each step, an unplaced component is selected, one at a time, and assigned to one of the unoccupied (free) positions. A position for the selected component, once occupied, remains in this state throughout the whole assignment process. The process is continued until all the components are assigned to the positions. The detailed template of the algorithm (in PASCAL-like form) is presented in Appendix 1 (Fig. A1).

2.2. *Solution Improvement*

The algorithms used for the initial solution improvement are based on the iterative (local) search methodology. The effectiveness of the iterative search depends on the neighbour-

hood structure chosen. The neighbourhood $N(\Pi \rightarrow 2^{\Pi})$ maps the set Π, i.e., the set of all the solutions, to the set 2^{Π}, i.e., the set of all the subsets of the set Π. A natural way by solving the quadratic assignment problem is to choose 2-exchange neighbourhood structure (pairwise interchanges). Given a solution (permutation) $p \in \Pi$, the 2-exchange neighbourhood N_2 is defined as follows

$$N_2(p) = \{p' | p' \in \Pi,\ \rho(p, p') = 2\}\,, \tag{2}$$

where $\rho(p, p')$ is the distance between the solutions p and p'. The distance is calculated according to the formula

$$\rho(p, p') = \sum_{i=1}^{n} \operatorname{sign} |p(i) - p'(i)|\,.$$

The solution p' in $N_2(p)$ can be obtained from the solution p by interchange of the positions of exactly two components. The size of the neighbour solutions set $N_2(p)$ is equal to $n(n-1)/2$, so the search process is quick enough.

The iterative algorithms differ, one from another, depending on how the search of the better solutions is performed. A large number of iterative algorithms are based on the greedy descent principle (e.g., Armour and Buffa, 1963; Hanan and Kurtzberg, 1972; Murtagh *et al*., 1982). The current solution is replaced by the new one from the neighbourhood (a move to another solution is performed) if only the difference of the objective function values ($\Delta_z = z(p') - z(p)$) is negative. The process is continued until no better solution exists in the neighbourhood. The last solution obtained is referred to as locally optimal solution. The greedy descent is a deterministic strategy, i.e., the decision to move to a new solution or not is predefined unambiguously by the current value of the objective function. The template of the greedy descent algorithm is presented in Appendix 2 (Fig. A2).

Another group of algorithms is composed of stochastic (non-greedy) algorithms. In this case, decisions are made according to some probability that depends on the difference of the objective function values. One of the most frequently used stochastic algorithms is simulated annealing (e.g., Burkard and Rendl, 1984; Wilhelm and Ward, 1987; Thonemann and Boelte, 1994). The algorithms based on the simulated annealing use the following decision rule (condition of updating). The probability of moving to the new solution in case of an increment of the value of the objective function is equal to $e^{-\Delta_z/t}$, where Δ_z is the difference of the objective function values, and t is a parameter, called temperature, to be lowered according to some cooling schedule. In case of a decrement of the objective function value, these algorithms work like the greedy algorithms. The detailed template of the simulated annealing (SA) algorithm (author's version) is presented in Appendix 3 (Fig. A3).

Thus, we have two large classes of the iterative search algorithms: greedy deterministic algorithms and non-greedy stochastic algorithms (see Table 1).

Table 1
Iterative algorithms classification scheme

Algorithms	Greedy	Non-greedy
Deterministic	**Greedy descent**	**Intensive search**
Stochastic	—	**Simulated annealing**

2.3. *Intensive Search Algorithm*

Our new iterative algorithm, intensive search algorithm, is based on the non-greedy strategy, but it is not a stochastic algorithm. It is a non-greedy deterministic algorithm (see Table 1). The key idea of the intensive search algorithm is that movies to the solutions, which increase (deteriorate) objective function value are permissible, in contrast to the greedy algorithms, where only the movies, which decrease objective function value are permissible. The intensive search algorithm uses the following decision rule: the current solution is replaced by the new one in the neighbourhood if the difference of the objective function values is negative or the difference is positive, but it is not greater than some range called threshold. Movies to the solutions, for which an increment of the objective function value is out of the range, are forbidden. The threshold τ is evaluated according to the formula

$$\tau = \left(\sum \Delta_z^+ \Big/ c^+ \right) \alpha, \tag{3}$$

where $\sum \Delta_z^+$ is the sum of all the positive differences of the objective function (the sum is being accumulated during the search process);

c^+ is the total number of the increments of the objective function value;

α is a coefficient (referred to as intensity level) within the interval $[0, 1]$.

(It is easy to observe that τ is the average objective function increment multiplied by the intensity level.)

Thus, the decision rule, in short, is defined as follows: the solution is to be updated if the inequality $\Delta_z < \left(\sum \Delta_z^+ / c^+ \right) \alpha$ is satisfied (Δ_z is the current difference of the objective function values). We avoid random element in the condition of updating in contrast to the simulated annealing (see Table 2). So, in case of the intensive search, the initial solution predetermines the final solution.

Table 2
Decision rules (conditions of updating)

Algorithms	Rules (conditions)
Greedy descent	$\Delta_z < 0$
Simulated annealing	$(\Delta_z < 0)$ OR $\left(r < e^{-\Delta_z / t} \right)$
Intensive search	$\Delta_z < \alpha \sum \Delta_z^+ \big/ c^+$

However, the intensive search possesses some features similar to those of the simulated annealing. Both intensive search and simulated annealing use special parameters to control optimization process. Namely, simulated annealing deals with the temperature (t), while the intensive search deals with the intensity level (α). We define intensity level to be inversely proportional to the number of the trials (permutations) already performed. Thus, the intensity level is high enough at the beginning of the search (large number of the increments of the objective function are permissible), it decreases slightly as the number of steps increases. It becomes 0 at the end of the search (only decrements of the objective function are available). The detailed template of the intensive search algorithm is presented in Fig. 1.

```
procedure intensive_search(p, n, ᾱ, K)
  /p – solution for the quadratic assignment problem, n – problem size (n ⩾ 2)/
  /ᾱ – initial value of the intensity level (ᾱ ∈ [0, 1])/
  /K – number of iterations of the intensive search/
  p˜ := p/p˜ – the best solution found so far/
  h_α := ᾱ/ max(1, 0.5Kn(n − 1) − 1) /intensity quantum/
  sum⁺ := 0 /sum of the positive differences of the objective function values/
  c⁺ := 0 /count of the positive differences of the objective function values/
  w := 1 /current trial number/
  for k := 1 to K do  /the main cycle of the intensive search/
     for i := 1 to n − 1 do
        for j := i + 1 to n do begin
           p′ := N2(p, i, j) /p′ – new solution in the 2-exchange neighbourhood/
           Δ_z := z(p′) − z(p) /current difference of the objective function values/
           if Δ_z < 0 then change:=TRUE
                      else begin
                        sum⁺ := sum⁺ + Δ_z, c⁺ := c⁺ + 1
                        α := Θ(ᾱ − (w − 1)h_α) /current intensity level/
                        if Δ_z < (sum⁺/c⁺)α then change:=TRUE
                                             else change:=FALSE
                      end /else/
           if change=TRUE then begin
             p := p′ /replace the current solution by the new one/
             if z(p) < z(p˜) then p˜ := p /update the best solution/
           end /if/
           w := w + 1 /next trial number/
        end /for/
  /end of the main cycle of the intensive search/
  return p˜
end intensive_search
```

Fig. 1. Template of the intensive search algorithm. In this template, Θ is an operator, e.g., $\Theta(x) = x$ or $\Theta(x) = x^2$ (we used the second one).

2.4. *Multistart Intensive Search Algorithm*

The input to the intensive search algorithm is: the initial intensity level $\bar{\alpha}$, the number of iterations K, as well as p, and n. Having p and K fixed ($K = \text{const}$), the optimization results ($p^{\tilde{}}$) depend only on the value of the initial intensity level $\bar{\alpha}$. For the distinct $\bar{\alpha}$, we get distinct final solutions $p^{\tilde{}}$, as well as distinct objective function values $z(p^{\tilde{}})$, i.e., z is a function of $\bar{\alpha}$ ($z = z(\bar{\alpha})$). The problem is that we do not know how exactly the initial value of the intensity level $\bar{\alpha}$ is to be set. In the other words, the goal is to find such a value $\bar{\alpha}$ that function $z(\bar{\alpha})$ is minimized.

Let us denote by $z_K(\bar{\alpha})$ the best value of the objective function obtained after K iterations for a fixed p and $\bar{\alpha}$. Then we get the following problem

$$\min_{0 \leqslant \bar{\alpha} \leqslant 1} z_K(\bar{\alpha}). \tag{4}$$

The function $z_K(\bar{\alpha})$ is stochastic, multimodal, as well as expensive (requires a lot of calculations). A simple technique is used for the optimization of this function. Some interval $[\bar{\alpha}_{\min}, \bar{\alpha}_{\max}] \subseteq [0, 1]$, called $\bar{\alpha}$ interval, is divided into $L-1$ subintervals using L points ($L \geqslant 2$). The coordinates of these points (i.e., values of the initial intensity level) are defined as follows

$$\bar{\alpha}^{(l)} = \bar{\alpha}_{\min} + (l-1)\delta, \tag{5}$$

where $l = 1, 2, \ldots, L$; $\delta = (\bar{\alpha}_{\max} - \bar{\alpha}_{\min})/(L-1)$ (i.e., δ is the distance between two neighbouring points).

The values of the objective function are calculated, i.e., calls to the intensive search procedure are made, at each point. In other words, L restarts of the intensive search procedure are performed. The solution p^* corresponding to the minimum value of the function $z_K(\bar{\alpha})$, obtained after L restarts, is regarded as an optimization result for the problem (1).

We entitled the above technique as a multistart intensive search (MIS) algorithm. The detailed template of the multistart intensive search algorithm is presented in Fig. 2.

3. Computational Experiments

We have carried out the computational experiments in order to compare the intensive search algorithm and other well-known algorithm, namely, simulated annealing. All the experiments were carried out by using the author's program "OPTIQAP" (OPTImizer for the QAP), developed to run on IBM PC computers.

We used the well-known problem instances taken from the quadratic assignment problem library (QAPLIB) compiled by Burkard *et al.* (1991). Computational results presented in Figures 3–10 have been obtained by using the instance NUG30.

In all the experiments, we used the initial solution generated by the initial solution construction procedure (see Fig. A1).

We made an emphasis on the following topics by carrying out the experiments.

procedure *multistart_intensive_search*(p, n, K)
  /p – solution for the quadratic assignment problem, n – problem size/
  /K – number of iterations of the intensive search ($K \geqslant 1$)/
  L =number of restarts
    (i.e., number of division points of the interval $[\bar{\alpha}_{\min}, \bar{\alpha}_{\max}]$) /$L \geqslant 2$/
  $p^* := p$/p^* – the best solution found so far/
  $z_{\min} := z(p)$/$z_{\min}$ – the minimum value of the objective function/
  $\delta := (\bar{\alpha}_{\max} - \bar{\alpha}_{\min}) \,/\, (L-1)$ /difference of the initial intensity level/
  for $l := 1$ **to** L **do begin** /the global cycle of the intensive search/
    $p' := p$ /initial solution is restored/
    $\bar{\alpha} := \bar{\alpha}_{\min} + (l-1)\delta$ /current initial intensity level/
    intensive_search$(p', n, \bar{\alpha}, K)$ /call to the intensive search procedure/
    if $z(p\prime) < z_{\min}$ **then begin**
      $p^* := p'$ /update the best solution/
      $z_{\min} := z(p')$ /update the minimum value of the objective function/
    end /if/
  end /for/
  return p^*
end *multistart_intensive_search*

Fig. 2. Template of the multistart intensive search algorithm.

3.1. *Choosing the Number of Iterations K*

The choice of the number of iterations can influence the quality of the results considerably. Firstly, if the number of iterations K is small, then no guarantee exists that the quality of the results is good enough (see Figs. 3, 4). Secondly, for large values of K, the results are better in many cases, but the number of trials as well as the computing time grows rapidly. In addition, sometimes the value of the objective function becomes worse even if the number of iterations increases (see Fig. 5), especially, with the small number of the restarts L. So, it is quite difficult to find right number of the iterations K (in fact, one more optimization problem arises, as an object of the further investigations).

We can fix K, for example, $K = 2^{\lfloor \log_2 n \rfloor + a}$ (a is an integer number between 0 and 10), or let it vary, suppose, between $K_{\min} = 1$ and $K_{\max} = 2^{\lfloor \log_2 n \rfloor + a}$. In the second case, the numbers of iterations K are chosen according to some choice rule. The choice rules are as follows: a) $K^{(m)} = (m-1)b + 1$ (arithmetical progression), b) $K^{(m)} = m^2$, c) $K^{(m)} = 2^{m-1}$ (geometrical progression), where $m = 1, 2, \ldots$, and b is a positive integer number.

3.2. *Choosing the Number of Restarts L*

Let us denote by $z_K(L)$ the best value of the objective function obtained after L restarts (calls to the intensive search procedure) for a fixed p and K. Then the obvious fact

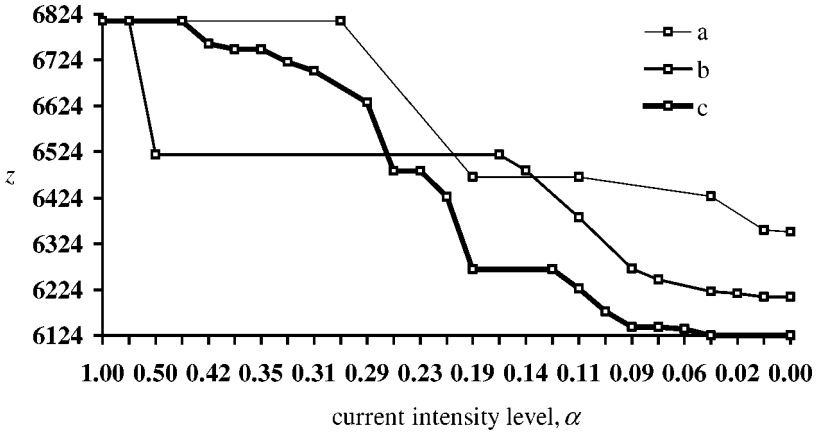

Fig. 3. The value of the objective function (z) versus current intensity level (α) and the number of iterations (K): a) $K = 9$ (minimum value of the objective function – 6350); b) $K = 27$ (6208); c) $K = 54$ (6124).

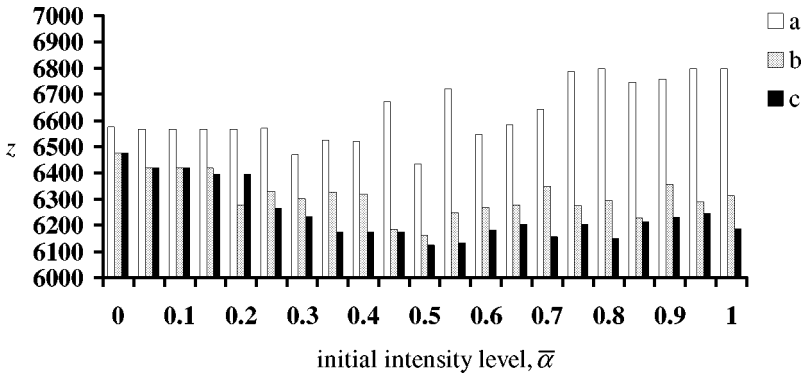

Fig. 4. The value of the objective function (z) versus initial intensity level ($\bar{\alpha}$) and the number of iterations (K): a) $K = 1$ (minimum value of the objective function – 6436); b) $K = 8$ (6162); c) $K = 59$ (6124).

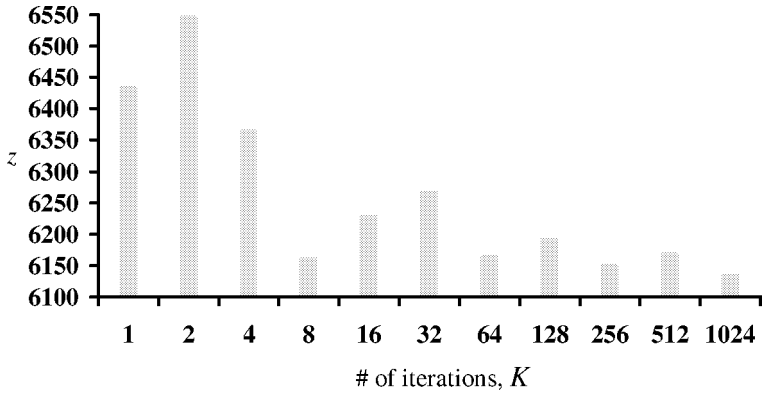

Fig. 5. The value of the objective function (z) versus the number of iterations (K). $\bar{\alpha} = 0.5$.

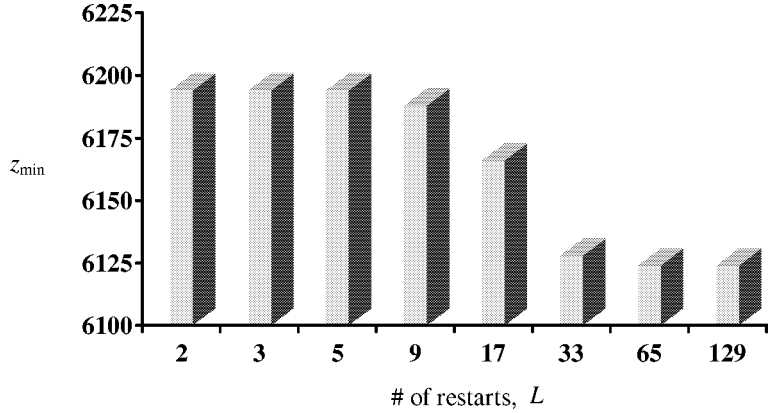

Fig. 6. Minimum of the values of the objective function ($z_{\min}$) versus the number of restarts (L). $z_{\min} = \min\limits_{1 \leqslant l \leqslant L} z_K(\bar{\alpha}^{(l)})$, where $\bar{\alpha}^{(l)}$ is calculated according to (5) formula ($\bar{\alpha}_{\min} = 0$, $\bar{\alpha}_{\max} = 1$). $K = 32$.

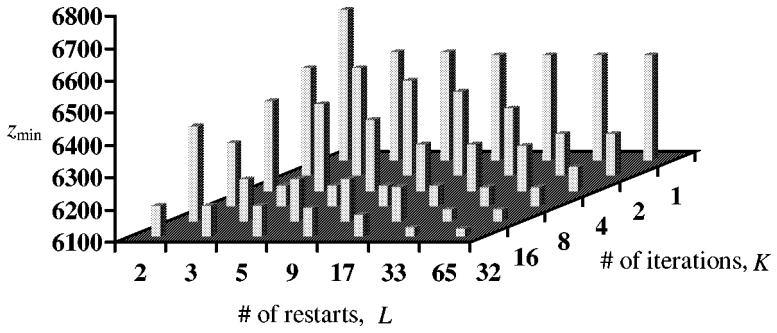

Fig. 7. Minimum of the values of the objective function ($z_{\min}$) versus the number of restarts (L) and the number of iterations (K). The overall minimum value of the objective function – 6124 ($L = 65$, $K = 32$).

holds. $z_K(2L-1) \leqslant z_K(L)$, where $L \geqslant 2$. The following simple strategy could be recommended: start with $L^{(1)} = 2$, then continue with $L^{(u+1)} = 2L^{(u)} - 1$, where $u = 1, 2, \ldots, u_{\max}$. We can choose $u_{\max}$ heuristically, for example, $u_{\max} = \lfloor 2 \log_2 n \rfloor$.

The illustrative examples of the correlation between number of restarts and the value of the objective function are presented in Figs. 6, 7.

3.3. *Dividing the $\bar{\alpha}$ Interval*

So far, we used the uniform dividing of the interval $[0, 1]$ ($\bar{\alpha}$ interval). Instead of the uniform dividing non-uniform dividing should be considered. The reason of such a strategy is that the "good" values of the objective function are, probably, not uniformly distributed along the interval $[0, 1]$ (see Fig. 8). For example, for the instance NUG30, we observed that, under certain conditions, the average of the objective function values in the interval $[0.5, 1]$ is less than that in the interval $[0, 0.5]$ (see Fig. 9).

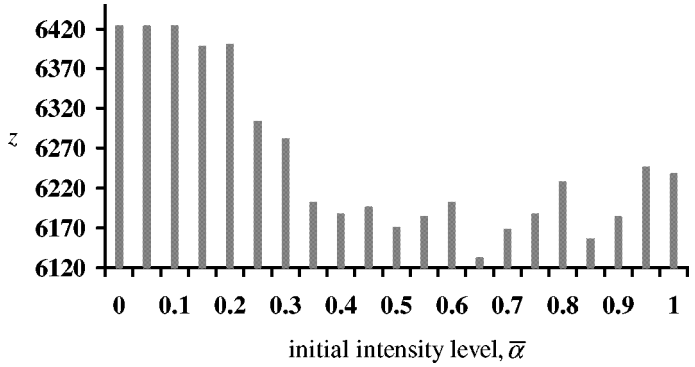

Fig. 8. The value of the objective function (z) versus initial intensity level ($\bar{\alpha}$).

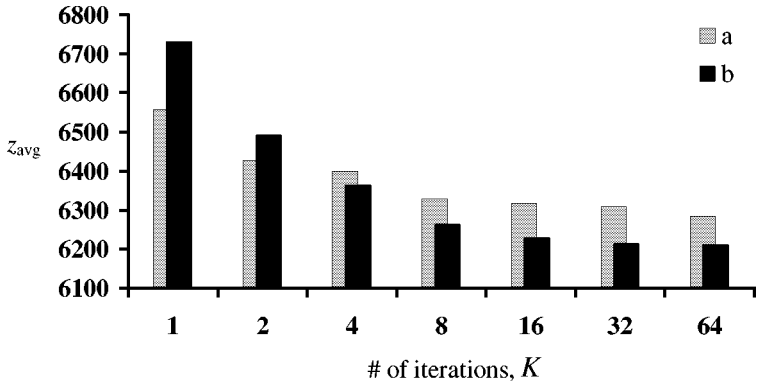

Fig. 9. Average of the values of the objective function (z_{avg}) versus the number of iterations (K). $z_{\text{avg}} = \sum_{l=1}^{L} z_K(\bar{\alpha}^{(l)}) \big/ L$, where $\bar{\alpha}^{(l)}$ is calculated according to (5) formula. a) $\bar{\alpha}_{\min} = 0$, $\bar{\alpha}_{\max} = 0.5$; b) $\bar{\alpha}_{\min} = 0.5$, $\bar{\alpha}_{\max} = 1$. $L = 9$.

3.4. *Reducing the $\bar{\alpha}$ Interval*

Preliminary results show that reducing of the $\bar{\alpha}$ interval is very promising idea (see Fig. 10). The concrete techniques for the reducing of the interval are the objects of the future investigations.

3.5. *Comparison of the Algorithms*

We compared intensive search and simulated annealing algorithms on several instances from QAPLIB, namely, NUG30, SKO42-90, STE36A, WIL50, and WIL100. We used to run the MIS procedure several times, with different numbers of restarts, each time. The number of restarts at uth run $L^{(u)}$ is given by $2^{u-1}+1$, where u is the current run number, $u = 1, 2, \ldots$ Really, we carry out only 2^{u-2} restarts at uth run ($u = 2, 3, \ldots$ ($L^{(1)} = 2$)) because we eliminate all the points previously processed. The number of iterations (K)

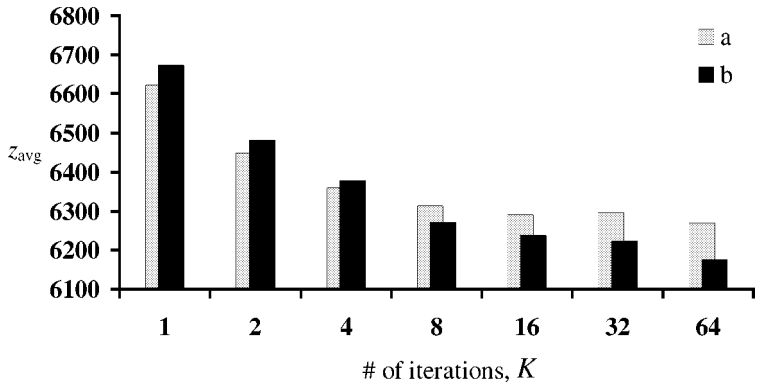

Fig. 10. Average of the values of the objective function (z_{avg}) versus the number of iterations (K): a) $\bar{\alpha} \in [0, 1]$ (interval length – 1); b) $\bar{\alpha} \in [0.5, 0.9]$ (length – 0.4). $L = 9$.

and the $\bar{\alpha}$ interval were chosen heuristically: $K = 2^{\lfloor \log_2 n \rfloor + 3}$; $\bar{\alpha} \in [0.4, 0.72]$ (interval length – 0.32), except the instance WIL100, for which $\bar{\alpha} \in [0.4, 0.56]$ (length – 0.16).

With the simulated annealing, we deal in the similar way, i.e., we perform many runs (restarts) of the SA procedure (see Fig. A3), except that initial temperature, t_0, is concerned. In fact, a temperature factor, τ, is varied. The number of iterations S is equal to $10K$, and the τ interval is $[0.1, 0.18]$ (except STE36A, for which $\tau \in [0.6, 0.68]$). In both cases, the runs are continued until the best known value of the objective function is found.

The comparison results are summarized in Table 3. In this table, we present values, at which the best known values (BKV) of the objective function have been obtained. The algorithms were tested on the 200MHz PENTIUM computer.

We also note that the IS procedure makes fewer computations per iteration, in comparison with the SA procedure, because expensive functions (exp and log) are not used.

4. Concluding Remarks

The quadratic assignment problem is very difficult optimization problem. In order to obtain satisfactory results in a reasonable time, heuristic algorithms are to be applied. One of them, the intensive search algorithm, is proposed in this paper.

Intensive search is a deterministic algorithm, in contrast to the most of the well-known algorithms (simulated annealing, genetic algorithms, greedy randomized adaptive search procedures). We regard this feature as main advantage of the proposed algorithm. Other advantages are:

– simple decision rule,
– easy programming and implementation,
– one initial solution is enough.

We also have to mention some shortages of the intensive search algorithm:

– the stopping criteria of the algorithm are difficult to determine,
– many input (control) parameters.

Table 3
Comparison of the algorithms

Instance name	BKV	IS			SA		
		# of runs	Total # of iterations[1]	Total search time, min:sec	# of runs	Total # of iterations[1]	Total search time, min:sec
NUG30	6124[3]	7	8320	0:40	5	21760	2:30
SKO42	15812[4]	5	4352	0:45	4	23040	5:40
SKO49	23386[4]	9	65792	25	6	84480	43
SKO56	34458[4]	11	262400	130	9	657920	320
SKO64	48498[4]	9	131584	100	2	15360	12
SKO72	66256[4]	9	131584	150	9	1315840	1660
SKO81	90998[5]	11	524800	700	11	5248000	>7000
SKO90	115534[4]	10	262656	650	∞[7]	∞	∞
STE36A	9526[4]	7	16640	2:40	4	23040	4:00
WIL50	48816[6]	8	33024	13	7	166400	70
WIL100	273038[5]	12	1049088	>3000	—	—	—

Remarks:
[1] the total number of iterations is given by $K(2^{M-1}+1)$, where M is the number of runs;
[2] the total number of iterations is given by $S(2^{M-1}+1)$;
[3] (Skorin-Kapov, 1990);
[4] (Taillard, 1991);
[5] (Fleurent and Ferland, 1994);
[6] (Thonemann and Boelte, 1994);
[7] BKV not found.

Nevertheless, the experiments carried out show that the intensive search algorithm gains advantage in most cases over other well-known algorithm, namely, simulated annealing (see Table 3).

The best known solutions were produced for all of the instances we tested. We expect to achieve even better results by applying new techniques of determination of the parameters $\bar{\alpha}$ and K. Effective algorithms for the solving optimization problem (4) should be applied, as well. All these things are subjects of the future papers.

Acknowledgement

Author is grateful to an anonymous referee for the remarks that helped to improve the initial version of this paper.

Appendix 1

procedure *initial_solution_construction*(p, n)
  /p – initial solution for the quadratic assignment problem, n – problem size/
  $p := (0, 0, \ldots, 0)$ /solution initialization/
  $i^1 := \arg \max\limits_{1 \leqslant i \leqslant n} \sum\limits_{k=1}^{n} f_{ik}$ /i^1 – index of the firstly selected component/
  $j^1 := \arg \min\limits_{1 \leqslant j \leqslant n} \sum\limits_{k=1}^{n} d_{jk}$ /j^1 – index of the position for the first component/
  $p(i^1) := j^1$ /assignment of the first component to the position/
  $I^1 := \{i^1\}$, $\bar{I}^1 := \{1, 2, \ldots, n\} \backslash I^1$, $J^1 := \{j^1\}$, $\bar{J}^1 := \{1, 2, \ldots, n\} \backslash J^1$
  for $q := 2$ **to** n **do begin**
    $i^q := \arg \max\limits_{i \in \bar{I}^{q-1}} \sum\limits_{k \in I^{q-1}} f_{ik}$ /i^q – index of the currently selected component/
    $j^q := \arg \min\limits_{j \in \bar{J}^{q-1}} \sum\limits_{k \in I^{q-1}} f_{i^q k} d_{jp(k)}$ /j^q – index of the position for
      the current component/
    $p(i^q) := j^q$ /assignment of the component to the position/
    $I^q := I^{q-1} \cup \{i^q\}$, $\bar{I}^q := \bar{I}^{q-1} \backslash \{i^q\}$
    $J^q := J^{q-1} \cup \{j^q\}$, $\bar{J}^q := \bar{J}^{q-1} \backslash \{j^q\}$
  end /for/
  return p
end *initial_solution_construction*

Fig. A1. Template of the algorithm for the initial solution construction.
The following notations are used:
i^q – the index of the selected component at step q;
j^q – the index of the position the component is assigned to at step q;
I^q – the set of indices of the components already placed after step q;
$\bar{I}^q$ – the set of indices of the unplaced components after step q;
J^q – the set of indices of the occupied positions after step q;
$\bar{J}^q$ – the set of indices of the unoccupied positions after step q.

Appendix 2

```
procedure greedy_descent(p, n)
  /p – current solution for the quadratic assignment problem, n – problem size/
  locally_optimal:=FALSE
  while locally_optimal≠TRUE do begin
      better_solution:=FALSE
      for i := 1 to n − 1 do
          for j := i + 1 to n do begin
              p' := N2(p, i, j) /p' – new solution in the 2-exchange neighbourhood/
              if z(p') < z(p) then begin
                  p := p' /replace the current solution by the new one/
                  better_solution:=TRUE
              end /if/
          end /for/
      if better_solution=FALSE then locally_optimal:=TRUE
  end /while/
  return p
end greedy_descent
```

Fig. A2. Template of the greedy descent algorithm.
In this algorithm, we denote by $N_2(p, i, j)$
the solution, which is obtained from the solution p
by interchanging the positions of components i and j.

Appendix 3

```
procedure simulated_annealing(p, n, τ)
  /p – solution for the quadratic assignment problem, n – problem size/
  /τ – temperature factor (τ ∈ [0, 1])/
  S :=number of iterations of the simulated annealing  /S ⩾ 1/
  p˜ := p /p˜ – the best solution found so far/
  t0 := τz(p)/n + 1 /initial temperature of the cooling schedule/
  t := t0  /current temperature/
  w := 3 /trial index/
  for s := 1 to S do   /the main cycle of the simulated annealing/
      for i := 1 to n − 1 do
          for j := i + 1 to n do begin
              p′ := N2(p, i, j)/p′ – new solution in the 2-exchange neighbourhood/
              Δz := z(p′) − z(p)  /current difference of the objective function values/
              if Δz < 0 then change:=TRUE
                      else begin
                        generate uniform random number r in [0, 1]
                        if r < e^(−Δz/t) then change:= TRUE else change:=FALSE
                      end  /else/
              if change=TRUE then begin
                p := p′  /replace the current solution by the new one/
                if z(p) < z(p˜) then p˜ := p  /update the best solution/
                t := t0/ log2 w
              end  /if/
              w := w + 1  /next trial index/
          end  /for/
    /end of the main cycle of the simulated annealing/
  return p˜
end simulated_annealing
```

Fig. A3. Template of the simulated annealing algorithm.

References

Armour, G.C., and E.S. Buffa (1963). Heuristic algorithm and simulation approach to relative location of facilities. *Management Science*, **9**, 294–309.

Bazaraa, M.S., and H.D. Sherali (1980). Bender's partitioning scheme applied to a new formulation of the quadratic assignment problem. *Naval Research Logistics Quarterly*, **27**, 29–41.

Burkard, R.E., S. Karisch and F. Rendl (1991). QAPLIB – a quadratic assignment problem library. *European J. of Operational Research*, **55**, 115–119.

Burkard, R.E., and F. Rendl (1984). A thermodynamically motivated simulation procedure for combinatorial optimization problems. *European J. of Operational Research*, **17**, 169–174.

Clausen, J., and M. Perregaard (1997). Solving large quadratic assignment problems in parallel. *Computational Optimization and Applications*, **8**(2), 111–127.

Fleurent, C., and J.A. Ferland (1994). Genetic hybrids for the quadratic assignment problem. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, **16**, 173–188.

Gilmore, P.C. (1962). Optimal and suboptimal algorithms for the quadratic assignment problem. *SIAM J. of Applied Mathematics*, **14**, 305–313.

Hanan, M., and J.M. Kurtzberg (1972). Placement techniques. In M.A. Breuer (Ed.), *Design Automation of Digital Systems: Theory and Techniques*, Vol. 1. Prentice-Hall, N.J., pp. 213–282.

Li, Y., P. Pardalos and M. Resende (1993). A greedy randomized adaptive search procedure for the quadratic assignment problem. *Tech. Report*, AT&T Bell Laboratories, Murray Hill, N.J., 22pp.

Mautor, T., and C. Roucairol (1994). A new exact algorithm for the solution of quadratic assignment problems. *Discrete Applied Mathematics*, **55**, 281–293.

Murtagh, B.A., T.R. Jefferson and V. Sornprasit (1982). A heuristic procedure for solving the quadratic assignment problem. *European J. of Operational Research*, **9**, 71–76.

Murthy, K.A., and P.M. Pardalos (1990). A polynomial-time approximation algorithm for the quadratic assignment problem. *Tech. Report CS–33–90*, The Pennsylvania State University.

Murthy, K.A., P.M. Pardalos and Y. Li (1992). A local search algorithm for the quadratic assignment problem. *Informatica*, **3**(4), 524–538.

Sherali, H.D., and P. Rajgopal (1986). A flexible polynomial time construction and improvement heuristic for the quadratic assignment problem. *Computers and Operations Research*, **13**(5), 587–600.

Skorin-Kapov, J. (1990). Tabu search applied to the quadratic assignment problem. *ORSA J. on Computing*, **2**(1), 33–45.

Steinberg, L. (1961). The backboard wiring problem: a placement algorithm. *SIAM Review*, **3**(1), 37–50.

Taillard, E. (1991). Robust taboo search for the QAP. *Parallel Computing*, **17**, 443–455.

Thonemann, U.W., and A.M. Boelte (1994). An improved simulated annealing algorithm for the quadratic assignment problem. *Working Paper*, University of Paderborn.

Wilhelm, M., and T. Ward (1987). Solving quadratic assignment problems by simulated annealing. *IEEE Trans*., **19**(1), 107–119.

A. Misevičius was born in 1962. He received the Dipl. Eng. degree from Kaunas Polytechnic Institute, Lithuania, in 1986. He got Doctor degree in the Kaunas University of Technology, in 1996. Now he is an Associate Professor at Practical Informatics Department of Kaunas University of Technology. His research interests include design and applications of optimization methods, computer-aided design.

Intensyviosios paieškos algoritmas kvadratinio paskirstymo uždaviniui

Alfonsas MISEVIČIUS

Straipsnyje aprašomas euristinis algoritmas, kuris pavadintas intensyviosios paieškos (IP) algoritmu ir išbandytas sprendžiant kvadratinio paskirstymo (KP) uždavinį. Nagrinėjamas taip pat šio algoritmo patobulinimas – daugkartinio starto intensyviosios paieškos (DSIP) algoritmas.

Siūlomas IP algoritmas priklauso negodžių deterministinių algoritmų klasei. IP algoritmo esmė slypi tame, kad skirtingai negu godžioje paieškoje, kur leidžiami "perėjimai" tik prie tų sprendinių, kurie pagerina (sumažina) tikslo funkcijos (TF) reikšmę, – čia leidžiama "pereiti" ir prie tų sprendinių, kurie pablogina tikslo funkcijos reikšmę. Intensyviosios paieškos atveju esamas sprendinys keičiamas nauju tada, kai TF reikšmių pokytis yra neigiamas arba tas pokytis yra teigiamas, bet neviršija tam tikro slenksčio. Slenkstis prilyginamas teigiamų TF pokyčių vidurkiui $\left(\sum \Delta_z^+ \big/ c^+\right)$, padaugintam iš tam tikro koeficiento (α), kuris vadinamas intensyvumo lygiu.

IP algoritmas primena žinomą atkaitinimo modeliavimo algoritmą, tačiau skiriasi nuo pastarojo deterministiniu veikimo pobūdžiu. Panašiai kaip atkaitinimo modeliavime, kur optimizavimo valdymui naudojamas temperatūros parametras, – IP algoritme naudojamas intensyvumo lygio parametras. Paieškos pradžioje intensyvumo lygis yra pakankamai aukštas, taigi leidžiamas didelis TF reikšmių pablogėjimų kiekis ("perėjimai" nuo vienų sprendinių prie kitų vykdomi intensyviai). Toliau šis lygis palaipsniui mažinamas, kol tampa lygus nuliui (galimi tik TF reikšmių pagerėjimai). Pradinė intensyvumo lygio reikšmė $(\bar{\alpha})$ parenkama iš intervalo [0, 1], o intensyvumo lygis mažinamas tam tikru žingsniu (h_α).

Naudojant IP algoritmą, tikslo funkcijos (z) reikšmė priklauso tik nuo pradinės intensyvumo lygio reikšmės $\bar{\alpha}$, kai kitos sąlygos vienodos. Taip KP uždavinys gali būti sprendžiamas minimizuojant realaus argumento funkciją $z(\bar{\alpha})$ ($\bar{\alpha} \in$[0, 1]). Šios funkcijos minimizavimui buvo išbandytas intervalo [0, 1] tolygaus dalijimo algoritmas – DSIP algoritmas.

Tiriant minėtus algoritmus, atlikta didelis kiekis eksperimentų, kuriuose pasinaudota KP uždavinio testiniais pavyzdžiais iš specialios bibliotekos – QAPLIB. Tyrimų rezultatai leidžia daryti išvadą, kad intensyvioji paieška, lyginant su godžia paieška, yra žymiai pranašesnė. Dar daugiau, gauti rezultatai liudija, jog IP algoritmas (kartu su DSIP algoritmu) daugeliu atveju pranoksta plačiai pripažintą atkaitinimo modeliavimo algoritmą.