KAUNAS UNIVERSITY OF TECHNOLOGY
VYTAUTAS MAGNUS UNIVERSITY
VILNIUS GEDIMINAS TECHNICAL UNIVERSITY

EVELINA STANEVIČIENĖ

# THE APPLICATION OF HYBRID GENETIC ALGORITHMS FOR THE GREY PATTERN PROBLEM

Summary of Doctoral Dissertation
Natural Sciences, Informatics (N 009)

2019, Kaunas

This doctoral dissertation was prepared at Kaunas University of Technology, Faculty of Informatics, Department of Multimedia Engineering during the period of 2013–2018.

**Editor:** Neringa Slapikevičiūtė

The official defence of the dissertation will be held at 2 p.m. on 26 August 2019 at the public meeting of Dissertation Defence Board of Informatics Science Field in Dissertation Defence Hall at Kaunas University of Technology.

Address: K. Donelaičio St. 73-403, 44249 Kaunas, Lithuania.
Tel. no. (+370) 37 300 042; fax. (+370) 37 324 144; e-mail doktorantura@ktu.lt.

The summary of the doctoral dissertation was sent on 26 July, 2019.
The doctoral dissertation is available on the internet http://ktu.edu and at the library of Kaunas University of Technology (K. Donelaičio St. 20, 44239 Kaunas, Lithuania) and Vytautas Magnus university (K. Donelaičio St. 58, 44248 Kaunas, Lithuania) and Vilnius Gediminas Technical University (Saulėtekio al. 11, 10223 Vilnius, Lithuania).

KAUNO TECHNOLOGIJOS UNIVERSITETAS
VYTAUTO DIDŽIOJO UNIVERSITETAS
VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

EVELINA STANEVIČIENĖ

# HIBRIDINIŲ GENETINIŲ ALGORITMŲ TAIKYMAS PILKŲJŲ ŠABLONŲ FORMAVIMO UŽDAVINIUI

Daktaro disertacijos santrauka
Gamtos mokslai, informatika (N 009)

2019, Kaunas

Disertacija rengta 2013–2018 metais Kauno technologijos universiteto Informatikos fakulteto Multimedijos inžinerijos katedroje.

**Mokslinis vadovas:**
Prof. dr. Alfonsas MISEVIČIUS (Kauno technologijos universitetas, gamtos mokslai, informatika, N 009).

**Redagavo:** Neringa Slapikevičiūtė

Disertacija bus ginama viešame informatikos mokslo krypties disertacijos gynimo tarybos posėdyje 2019 m. rugpjūčio 26 d. 14.00 val. Kauno technologijos universiteto Disertacijų gynimo salėje.

Adresas: K. Donelaičio g. 73-403, 44249 Kaunas, Lietuva.
Tel. +370 37 300 042; faks. +370 37 324 144; el. paštas doktorantura@ktu.lt.

Disertacijos santrauka išsiųsta 2019 m. liepos 26 d.
Su disertacija galima susipažinti interneto svetainėje http://ktu.edu ir Kauno technologijos universiteto (K. Donelaičio g. 20, 44239 Kaunas), Vytauto Didžiojo universiteto (K. Donelaičio g. 58, 44248 Kaunas) ir Vilniaus Gedimino technikos universiteto (Saulėtekio al. 14, 10223 Vilnius) bibliotekose.

## 1. INTRODUCTION

Optimization methods and algorithms is a relevant area of computer science. A significant group of these methods consists of modern heuristic algorithms, including hybrid genetic algorithms.

Modern heuristic algorithms have been successfully developed to solve complex theoretical-mathematical and practical tasks. An example of such a task is the quadratic assignment problem and its particular case – the grey pattern problem (GPP).

The focus is on the application of heuristic algorithms in the creation of the superior-quality colour patterns (digital colour halftone textures). The creation of the digital colour halftones is based on the grey pattern problem, where the goal is to obtain the finest (optimal) grey pattern of a pre-defined density. The aims of the dissertation are theoretical—to make theoretical prototypes of effective heuristic algorithms designed for the class of complex theoretical combinatorial optimization tasks, —as well as practical—to adapt prototypes for a specific practical-technical task.

Various types of heuristic algorithms can be applied to the given task (GPP), starting with the local search algorithms and ending with the biological process-inspired simulation algorithms. An important aspect is related to hybridization, which means that the goal is to achieve beneficial synergy of combining different algorithms. A good example of this is a combination of fast single-solution based heuristics (on the one side) and population-based (evolutionary) algorithms (on the other side). In addition to combining multiple heuristics, another effective strategy is using a set of various search algorithms.

**The Object of the Research**. The object of the research is the hybrid genetic algorithms for the grey pattern problem.

**The Aim of the Research**. The aim of the research is to investigate the effectiveness of hybrid genetic algorithm modifications for the grey pattern problem and to determine the most effective algorithm variants.

**The Tasks of the Research**. To achieve the aim, the following tasks were outlined:

1. Exploratory study of existing algorithmic solutions for the grey pattern problem.
2. Proposal and implementation of a new hybrid genetic algorithm for the grey pattern problem.
3. Experimental investigation of algorithm efficiency and analysis of the obtained results.

**The Methods of the Research**. The research methodology is based on the use of heuristic and metaheuristic combinatorial optimization algorithms and the analysis of their efficiency.

**Scientific Novelty and Practical Relevance**. Scientific novelty is characterized by two aspects.

1. A hybrid genetic algorithm of an improved structure (architecture) was proposed for the grey pattern problem.
2. The hybrid genetic algorithm was implemented with a new type of integrated iterative tabu search procedure combined with the efficient adaptive perturbation of solutions.

Hybrid genetic algorithm (HGA) was successfully applied to the grey pattern quadratic assignment problem, which is utilized as a formal model for construction of the digital halftones.

**Approbation of the Research Results**. The main results of the dissertation were published in 6 scientific publications: 2 in the journals included in the list of scientific international databases (Indexed in the Web of Science with Impact Factor). The results were also presented at 3 international conferences.

**The structure and Volume of the Dissertation**. The dissertation consists of an introduction, 3 main chapters, conclusions, a list of references, a list of the author's publications and 2 appendices. The total volume of the dissertation is 88 pages, including 38 figures, 30 tables and 93 references.

## 2. HYBRID GENETIC ALGORITHM AND ITS DIFFERENT MODIFICATIONS

### 2.1 Grey pattern quadratic assignment problem

The grey pattern quadratic assignment problem is a special case of a well-known combinatorial optimization problem, the quadratic assignment problem (QAP) [1]. GPP can be formulated as follows: given two matrices $\boldsymbol{A} = (a_{ij})_{n\times n}$ and $\boldsymbol{B} = (b_{kl})_{n\times n}$ and the set $\Pi_n$ of permutations of the integers from 1 to $n$, find a permutation $p \in \Pi_n$ that minimizes:

$$z(p) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} b_{p(j)p(i)}; \tag{1}$$

where $a_{ij} = 1$ for $i, j = 1, \dots, m$ $(1 \le m < n)$ and $a_{ij} = 0$ otherwise. In this way, the objective is changed to finding the permutation elements $p(1), \dots, p(m)$ ( $1 \le p(i) \le n, i = 1, \dots, m$) such that the simplified function $z(p)$ is minimized:

$$z(p) = \sum_{i=1}^{m} \sum_{j=1}^{m} b_{p(i)p(j)}. \tag{2}$$

In the context of the GPP, the values $b_{kl}$ are defined according to the following rule (see also [2]):

$$b_{kl} = b_{(r-1)n_2+t \quad (u-1)n_2+v} = \omega_{rtuv} \ , \tag{3}$$

$$\omega_{rtuv} = \min_{w_1, w_2 \in \{-1,0,1\}} \frac{1}{(r-u+w_1 n_1)^2 + (t-v+w_2 n_2)^2} \ ; \tag{4}$$

where $r, t = 1, \dots, n_1$, $s, u = 1, \dots, n_2$, $n_1 \times n_2 = n$.

We have a grid of dimensions $n_1 \times n_2$. More precisely, we have $n = n_1 \times n_2$ squares (points) in the grid: there are $m$ black squares and $n-m$ white squares. (Other colours may be considered instead of black and white.) This forms a grey (or colour) pattern of density $m/n$. The aim is to have a grey (colour) pattern where the black points (colour points) are distributed in the most uniform possible way.

The elements of the solution found determine the locations in the grid, where the black squares have to be placed. The coordinates $r$ and $t$ of the black squares are obtained according to these formulas:

$$r = \lfloor (p(i) - 1)/n_2 \rfloor + 1, \tag{5}$$

$$t = \big((p(i) - 1) mod\ n_2\big) + 1\ ; \tag{6}$$

where $i \leq m$.

For example, we have permutation $p = (2, 6, 7, 1, 5, 3, 4, 9, 8)$, $n_1 = 3$, $n_2 = 3$, $n = 9$, $m = 3$, then, the following grey pattern is obtained (see Fig. 1).

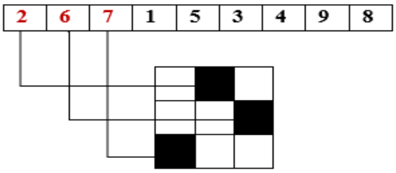

**Figure 1.** A graphical illustration of correspondence of the analytical solution to the graphical image

## 2.2 Basic definitions

**Definition 1.** A neighbourhood function $\Theta: \Pi_n \to 2^{\Pi_n}$ assigns for each $p \in \Pi_n$ a set $\Theta(p) \subseteq \Pi_n$ — the set of neighbouring solutions of $p$. With the permutation-based problems, a common practice is to use the 2-exchange neighbourhood function $\Theta_2$ which is defined in the following way: $\Theta_2(p) = \{p' : p' \in \Pi_n, \delta_H(p, p') = 2\}$, where $\delta_H(p, p')$ is the Hamming distance between the permutations $p$ and $p'$. (The Hamming distance between two permutations $p_1$ and $p_2$ can be declared as $\delta_H(p_1, p_2) = |\{i : p_1(i) \neq p_2(i)\}|$.)

**Definition 2.** The 1-interchange neighbourhood function $\Theta_1$ is defined in such a way that every neighbouring solution $p' \in \Theta_1(p)$ is obtained from the current solution $p$ by simply interchanging one element of $\{p(i) : i = 1, \dots, m\}$ with another element of $\{p(j) : j = m + 1, \dots, n\}$. This neighbourhood function maintains solution feasibility:

$$\Theta_1(p) = \{p' : p' \in \Pi_n, \delta(p, p') = 1\}\ ; \tag{7}$$

where $\delta$ denotes the distance between solutions.

**Definition 3.** The distance between two GPP solutions $p_1$ and $p_2$ can be defined in the following way:

$$\delta(p_1, p_2) = m - |\{p_1(i): i = 1, \dots, m\} \cap \{p_2(i): i = 1, \dots, m\}|; \qquad (8)$$

where $0 \le \delta \le m$, $\delta(p, p) = 0$, $\delta(p_1, p_2) = \delta(p_2, p_1)$. Let $p(v)$ $(v = 1, \dots, m)$ and $p(w)$ $(w = m + 1, \dots, n)$ be two items to be swapped. Then, a short notation of the form $p^{v,w}$ can be used in such a way that $p^{v,w}(i) = \begin{cases} p(i), & i \ne v, w \\ p(v), & i = w \\ p(w), & i = v \end{cases}$. This means that $p^{v,w}$ is obtained from $p$ by interchanging the items $p(v)$ and $p(w)$ ($p$ is said to move to $p^{v,w}$). Then, $\delta(p, p^{v,w}) = 1$ , $p^{v,v} = p$, $(p^{v,w})^{v,w} = p$.

**Definition 4.** The difference of the objective function is calculated according to this formula:

$$\Delta(p^{v,w}, p) = z(p^{v,w}) - z(p) = 2\Big(c\big(p(w)\big) - c\big(p(v)\big) - B\big(p(v), p(w)\big)\Big); \qquad (9)$$

where $c(x)$ is a contribution (a sum of related distances) of the element $x$.

**Definition 5.** The sum of related distances is calculated according to the following formula:

$$c(x) = \sum_{y=1}^{m} b_{xp(y)}; \qquad (10)$$

where $x = 1, \dots, n$.

**Definition 6.** After the exchange, the contributions are updated according to the expression:

$$c(x) = \begin{cases} c(x) + B\big(x, p(v)\big), & x = p(w) \\ c(x) - B\big(x, p(w)\big), & x = p(v) \\ c(x) + B\big(x, p(v)\big) - B\big(x, p(w)\big), & x \ne p(v), x \ne p(w) \end{cases}; \qquad (11)$$

where $x = 1, \dots, n$.

The neighbourhood $\Theta_2$ is defined according to the following formula (see also [3]):

$$\Theta_2(p) = \left\{ p^{\nabla\nabla} \middle| \begin{array}{c} p^{\nabla\nabla} = p^{ijrt}, p^{\nabla\nabla} \in \Pi_n, i = 1, \dots, m, j = m + 1, \dots, n, \\ r = (i \text{ modulo } m) + 1, t = \max((j \text{ modulo } n) + 1, m + 1) \end{array} \right\}; \qquad (12)$$

where $p^{ijrt}(k) = \begin{cases} p(k), k \ne i, j, r, t \\ p(i), k = j \\ p(j), k = i \\ p(r), k = t \\ p(t), k = r \end{cases}$, $k = 1, \dots, n$.

$\Theta_3$is defined as follows:

$$\Theta_3(p) = \left\{ p^{\nabla\nabla\nabla} \middle| \begin{array}{l} p^{\nabla\nabla\nabla} = p^{ijrtuv}, p^{\nabla\nabla\nabla} \in \Pi_n, i = 1, \ldots, m, j = m+1, \ldots, n, \\ r = (i \text{ modulo } m) + 1, t = \max((j \text{ modulo } n) + 1, m + 1), \\ u = (r \text{ modulo } m) + 1, v = \max((t \text{ modulo } n) + 1, m + 1) \end{array} \right\}; \quad (13)$$

where $p^{ijrtuv}(k) = \begin{cases} p(k), k \neq i, j, r, t, u, v \\ p(i), k = j \\ p(j), k = i \\ p(r), k = t \\ p(t), k = r \\ p(u), k = v \\ p(v), k = u \end{cases}$, $k = 1, \ldots, n$.

When using the neighbourhood $\Theta_2$, the difference $\Delta z(p, p^{ijrt})$ is equal to:

$$\Delta z(p, p^{ijrt}) = 2\begin{pmatrix} c_{p(j)} - c_{p(i)} + c_{p(t)} - c_{p(r)} + b_{p(i)p(r)} + b_{p(j)p(t)} - \\ b_{p(i)p(j)} - b_{p(i)p(t)} - b_{p(r)p(j)} - b_{p(r)p(t)} \end{pmatrix}. \quad (14)$$

Using the neighbourhood $\Theta_3$, the difference $\Delta z(p, p^{ijrtuv})$ is equal to:

$$\Delta z(p, p^{ijrtuv}) = 2\begin{pmatrix} c_{p(j)} - c_{p(i)} + c_{p(t)} - c_{p(r)} + c_{p(v)} - c_{p(u)} + \\ b_{p(i)p(r)} + b_{p(i)p(u)} + b_{p(r)p(u)} + b_{p(j)p(t)} + b_{p(j)p(v)} + \\ b_{p(t)p(v)} - b_{p(i)p(j)} - b_{p(i)p(t)} - b_{p(i)p(v)} - b_{p(r)p(j)} - \\ b_{p(r)p(t)} - b_{p(r)p(v)} - b_{p(u)p(j)} - b_{p(u)p(t)} - b_{p(u)p(v)} \end{pmatrix}. \quad (15)$$

**Definition 7.** The GPP solution $\overline{p} \in \Pi_n$ is an opposition-based solution with respect to the solution $p$ if $\delta(p, \overline{p}) = m$.

**Definition 8.** The GP solution $p^{\maltese} \in \Pi_n$ is a backbone solution (with respect to two underlying solutions $p_1, p_2$) if simultaneously $\delta(p^{\maltese}, p_1) \leq \lceil m/2 \rceil$ and $\delta(p^{\maltese}, p_2) \leq \lceil m/2 \rceil$ (see also [4]).

## 2.3 Description of the hybrid genetic algorithm for the grey pattern problem

The new algorithm (HGA) (see also [5]) is based on the hybrid genetic algorithm framework, where the population-based evolutionary search is combined with the local improvement of the produced offspring.

The algorithm starts with the generation of the initial population (of fixed size $PS$). All the population members undergo local improvement (optimization). The genetic algorithm then performs generations until the pre-defined number of generations, $N_{gen}$, is completed. At each generation, the standard genetic operations—selection, crossover, population replacement—take place (without the direct involvement of mutation). Every new solution (offspring) produced by the crossover operator is subject to local improvement. To preserve the diversity of the population members, an enhanced population replacement strategy is adopted. Then, a population restart process is incorporated to renew the population if the genetic variability is lost and/or a number of idle generations exceed the predefined limit.

The high-level description of the hybrid genetic algorithm is presented in Figure 2.

```
procedure Hybrid_Genetic_Agorithm;
//input: $n$, $m$, $B$
//output: $p^*$ – the best solution found
//parameters: $PS$ – population size, $N_{gen}$ – number of generations,
//      $N_{offspr}$ – number of offspring per generation, $DT$ – distance threshold, $L_{idle\_gen}$ – idle generations limit,
//      control parameters for local optimizer (hierarchical iterated tabu search algorithm) (see Table 1)
begin
  get data, parameters;
  initialize algorithm variables;
  generate sorted INITIAL POPULATION $P$ of size $PS$;
  $p^* := \underset{p \in P}{\operatorname{argmin}}\{z(p)\}$; //memorize the best solution of the initial population
  $gen\_index := 1$;
  while $gen\_index <= N_{gen}$ do begin
    for $i := 1$ to NUMBER OF OFFSPRING PER GENERATION do begin
      perform SELECTION procedure to choose the parents $p', p'' \in P$;
      produce offspring by applying CROSSOVER procedure to $p', p''$;
      apply LOCAL IMPROVEMENT to the produced offspring $p^\circ$,
      get an (improved) solution $p^☆$ (and $p^{☆☆}$);
      if $z(p^☆) < z(p^*)$ (or $z(p^{☆☆}) < z(p^*)$) then $p^* := p^☆$ (or $p^* := p^{☆☆}$);
       //memorize the best found solution
      add $p^*$ to a pool of offspring
      endfor;
      if number of idle generations exceeds the predefined limit
       $L_{idle\_gen}$ then begin
         POPULATION RESTART; //perform population restart procedure
        if $\min_{p \in P}\{z(p)\} < z(p^*)$ then $p^* := \underset{p \in P}{\operatorname{argmin}}\{z(p)\}$
      end //of if
     else perform POPULATION REPLACEMENT; //update and sort the current population
    $gen\_index := gen\_index + 1$ //go to the next generation
  endwhile
end.
```

**Note.** There also are several control parameters for the local optimizer (the hierarchical iterated tabu search algorithm) (for more details, see Section 2.4 and Table 1).

**Figure 2.** Description of the hybrid genetic algorithm

### 2.3.1 Initial population construction

The initial population is generated as follows.

1. Let $flag = $ 'OFF', $P = \emptyset$, $k = 1$, $l = 1$. ($l$ is the lexicographic index of the generated solution.)

2. If $flag = $ 'OFF' then generate a random permutation (solution) $p_1$; otherwise, generate an opposition-based random permutation $p_l$. $l = l + 1$.

3. Apply the hierarchical iterated tabu search algorithm to the generated solution and get the improved solution $p_l^*$.

4. If $(flag = \text{'OFF'})$ and $(k = 1)$ then: a) include the solution $p_1^*$ into the population $P$; b) $flag = \text{'ON'}$; c) go to step 2.

5. If $(flag = \text{'ON'})$ and $(k = 1)$ and $(z(p_l^*) < z(p){:}\, p \in P)$ then: a) replace the 1st member of the population by the solution $p_l^*$ ; b) go to step 2.

6. If $\left((z(p_l^*) \neq z(p){:}\, \forall p \in P) \text{ and } \left(\min\limits_{p \in P}\{\delta(p_l^*, p)\} \geq DT\right)\right)$ or $\left(z(p_l^*) < \min\limits_{p \in P}\{z(p)\}\right)$, then include the solution $p_l^*$ into the population $P$. Otherwise, include the random solution $p_l$ into the population $P$.

7. $k = k + 1$. If $k \leq PS$ then go to step 2; otherwise, the initial population formation is finished.

### 2.3.2 Parent selection

At each generation, two solutions (permutations) $p'$ and $p''$ are selected in the population $P$ to serve as parents for reproduction.

### 2.3.3 Crossover operator

The goal of the crossover operator is to produce an offspring from a pair of parents. The principle of functioning of the used crossover is based on two concepts: a backbone solution and an opposition-based solution (see also [4, 5]).

This allows both, to preserve the common elements (genes) in two selected parents and to introduce completely new genes. The so-called greedy adaptive procedure (GAP) is applied for this purpose. Thus, two offspring solutions are generated: the optimized offspring solution and the opposite offspring solution. Some specific details are as follows.

The values of the short-term array (gene frequencies) $f_{ST}^{cross}$ are calculated by this expression:

$$f_{ST}^{cross}(i) = \left|\{i{:}\, i \in \{p'(k){:}\, k = 1, \dots, m\}, i \in \{p''(k){:}\, k = 1, \dots, m\}\}\right|, \quad (16)$$

where $i = 1, \dots n$, $p'$, $p''$ are the corresponding parental solutions. The $m$ genes with the highest frequency are chosen to form the backbone solution $p^{✢}$. After this, the greedy adaptive procedure is applied, which respects only $m/2$ genes with the largest frequency. So, the GAP receives a partial solution $p^{✢}$ (the elements $p^{✢}(1)$, …, $p^{✢}(m/2)$) as an input. The GAP chooses the element, one at a time, and adds it to the current partial solution. In particular, GAP adds, at each iteration $q$ $(q = 1, \dots, m/2)$, the element from the set of unselected elements $\{j{:}\, j = 1, \dots, n\} \setminus \{p(i){:}\, i = 1, \dots, m/2 + q - 1\}$ with the minimum contribution value (see formula (7)) across all the unselected elements, i.e. $j = \operatorname{argmin}_{j \in \{j:j=1,\dots,n\} \setminus \{p(i):i=1,\dots,m/2+q-1\}} \{c(p(j))\}$. This continues until the solution is completed.

For the generation of the opposite (opposition-based) solution, a long-term frequency array $f_{LT}^{cross}$ is used. The initialization of $f_{LT}^{cross}$ is done before running the genetic algorithm. The values of $f_{LT}^{cross}$ are updated each time the new backbone solution is constructed, i.e. $f_{LT}^{cross}(p^{\circ}(i)) = f_{LT}^{cross}(p^{\circ}(i)) + 1$, where $p^{\circ}$ is the backbone solution.

### 2.3.4 Hierarchical iterated tabu search algorithm

The hierarchical iterated tabu search algorithm proposed by me follows the hierarchical iterated local search paradigm [6]. In the case of the tabu search (TS), first the iterated tabu search—ITS—is obtained by combining the tabu search and some perturbations. Further, the ITS algorithm itself is combined with another ITS algorithm, which results in the "ITS-ITS" algorithm. Each copy contains three main ingredients: 1) iterated tabu search procedure; 2) candidate acceptation; 3) perturbation procedure.

*A. Tabu search.*

The tabu search procedure plays an essential role in the hierarchical ITS algorithm. In the simplest way, the TS procedure uses the 1-exchange neighbourhood $\Theta_1$. TS starts with the current solution and iteratively swaps an element of the set $M = \{p(i): i = 1, \dots, m\}$ with an element of the set $N = \{p(i): i = m + 1, \dots, n\}$ in such a way that the objective function value is minimized by taking into account the tabu condition and aspiration criterion.

To reduce the computational time, the modified neighbourhood $\Theta_1^*$ is used which is defined as follows (see also [7]):

$$\Theta_1^*(p) = \{p' : p' \in \{p(i): i = 1, \dots, m\} \setminus \{p(v)\} \cup \{p(w)\}, p(v) \in M', p(w) \in N'\}. \quad (17)$$

The sets $M'$, $N'$ are formed in the following way:

$$M' = \{p(i): c(p(i)) \geq threshold_1,\ i = 1, \dots, m\}, \quad (18)$$

$$N' = \{p(i): c(p(i)) \leq threshold_2,\ i = m + 1, \dots, n\}; \quad (19)$$

where $c$ is the contribution array, $threshold_1 = \max\{c(p(i)): i = 1, \dots, m\} - \rho BMax$, $threshold_2 = \min\{c(p(i)): i = m + 1, \dots, n\} + \rho BMax$, $BMax = \max\{b_{kl}: k = 1, \dots n, l = 1, \dots n\}$, $\rho$ ($\rho > 0$) is a parameter (a neighbourhood size factor).

The tabu list $TabuList$ is organized as a matrix, where the tabu list entry $TabuList(p(v), p(w))$ stores the current iteration number plus the tabu tenure $h$, i.e. the number of the iteration starting at which the corresponding elements $(p(v), p(w))$ may again be interchanged. The interchange of elements $p(v), p(w)$ is not allowed if the value of $TabuList(p(v), p(w))$ is equal or greater than the current iteration number. The tabu status is ignored if the aspiration criterion is met, i.e. the interchange results in a solution that is better than the best solution so far.

In addition to the tabu list, a long-term memory like mechanism to maintain an archive of good solutions that were evaluated but not chosen [8] is also used. The goal is to diversify the search process and explore more regions of the search space. To implement this mechanism, an archive ($Archive$) is used, which is composed of the so-called “second” solutions.

*B. Iterated tabu search.*

The TS procedure transforms the current solution into the optimized solution. Perturbation is applied to a chosen optimized candidate solution that is selected by a defined candidate acceptation rule. Candidate acceptation can be implemented in many ways. The so-called “where-you-are” rule is used, which means that the last found improved solution is always chosen. The perturbed solution serves as an input for the TS procedure, which starts immediately after the perturbation procedure is executed. The perturbation procedure is very simple in its structure and it consists of two parts: a) random mutation and b) re-construction of the mutated solution by a fast greedy adaptive procedure. First, the accepted candidate solution undergoes a random mutation process; in particular, the solution is “disintegrated” by disregarding (removing) $\mu$ elements from the current solution ($\mu$ is a parameter called the mutation rate). The $\mu$ elements are chosen in a random way. The value of $\mu$ is relatively small in our algorithm ($\mu = \lfloor 0.15m \rfloor$), so only a minor fraction of elements is involved in the mutation procedure. Second, the mutated partial solution is subject to re-construction by the fast-greedy adaptive procedure (FGAP), which is identical to that used in the crossover operator, except that a more effective calculation of the contributions is applied.

TS again returns an improved solution. This solution (or possibly some other previously optimized solution), in turn, is perturbed, and so on. The best-found solution is regarded as the resulting solution of ITS. The overall process continues until a pre-defined number of iterations is performed.

*C. Hierarchical iterated tabu search.*

The 1-level hierarchical iterated tabu search (1-HITS) algorithm can be obtained from the ITS algorithm. The structure of the algorithm remains practically unchanged, except that the ITS algorithm (instead of the TS algorithm) is used for the solution improvement.

It is possible to further extend the 1-HITS algorithm in a very gentle way. New extension is entitled as 2-HITS. The pseudo-code of 2-HITS is almost identical to the one of 1-HITS, except that the invocation of the ITS procedure is substituted by the invocation of the 1-HITS procedure.

### 2.3.5 Population management

After the offspring is improved by HITS, the new solution ($p^{☆}$) is tested if it differs from other solutions in a population. If it is the case, the new solution is checked to determine if it is better than the best population solution, or if the distance between the new solution and the population ($\delta(p^{☆}, P) = \min_{p \in P}\{\delta(p^{☆}, p)\}$) is greater than or equal to the distance threshold $DT$. If this is true, then the new solution replaces the worst solution in the current population ($P = P \cup \{p^{☆}\} \setminus \{p_{worst}\}$, where $p_{worst} = \underset{p \in P}{\operatorname{argmin}}\{z(p)\}$). Otherwise, the population remains unaltered and the algorithm continues with the next generation. This rule is created to maintain both the high-quality and sufficient diversity of the members of population.

### 2.3.6 Restart

The restart of the genetic algorithm takes place if the solutions of the population are not improved for $L_{idle\_gen}$ generations ($L_{idle\_gen}$ is an idle generations limit, which is set to $\lfloor 0.15N_{gen} \rfloor$, $N_{gen}$ is the number of generations). The restart is performed by simply constructing a new population (see Section 2.3.1).

## 2.4 Components of the hybrid genetic algorithm

The following components (features) of the hybrid genetic algorithm were investigated (see also [9]).

The used values of the control parameters of HGA are presented in Table 1.

**Table 1.** Default values of the control parameters of the hybrid genetic algorithm

| **Parameter** | **Value** | **Remarks** |
|---|---|---|
| Population size, $PS$ | 20 | |
| Number of generations, $N_{gen}$ | 100 | |
| Idle generations limit, $L_{idle\_gen}$ | $\lfloor 0.03N_{gen} \rfloor$ | $0 < L_{idle\_gen} \leq N_{gen}$ |
| Distance threshold, $DT$ | $\lfloor 0.15m \rfloor$ | $0 \leq DT \leq m$ |
| Number of hierarchical iterated tabu search iterations, $Q_{HIER}$ | 256 | $Q_{HIER} = Q \times Q_1 \times Q_2 \times Q_3 \times Q_4 \times Q_5 \times Q_6 \times Q_7$[†] |
| Number of initial hierarchical iterated tabu search iterations, $Q_{IHIER}$ | 4096 | $Q_{IHIER} = Q \times Q_1 \times Q_2 \times Q_3 \times Q_8 \times Q_9 \times Q_{10} \times Q_{11}$[††] |
| Number of restart hierarchical iterated tabu search iterations[†††], $Q_{RHIER}$ | | $Q_{RHIER} = Q \times Q_1 \times Q_2 \times Q_3 \times Q_{12} \times Q_{13} \times Q_{14} \times Q_{15}$ |
| Number of tabu search iterations, $\tau$ | 50 | |
| Tabu tenure, $h$ | $\lfloor 0.3m \rfloor$ | $h > 0$ |
| Idle iterations limit, $L_{idle\_iter}$ | $\lfloor 0.2\tau \rfloor$ | $0 < L_{idle\_iter} \leq \tau$ |
| Neighbourhood size factor for tabu search, $\rho$ | 0.4 | $\rho > 0$ |
| Randomization coefficient for tabu search, $\alpha$ | 0.02 | $0 < \alpha < 1$ |
| Mutation rate for hierarchical iterated tabu search, $\mu$ | $\lfloor 0.15m \rfloor$ | $0 < \mu < m$ |

[†] $Q = Q_1 = Q_2 = Q_3 = Q_4 = Q_5 = Q_6 = Q_7 = 2$;
[††] $Q_8 = Q_9 = Q_{10} = Q_{11} = 4$ ($Q_{IHIER} = 16Q_{HIER}$);
[†††] $Q_{12} = Q_{13} = Q_{14} = Q_{15} = 3$.

### 2.4.1 Component "INITIAL POPULATION"

The initial population component is the component that determines the way in which the initial (starting) population of solutions (individuals) is constructed. For the particular variants of components, short notations like **`IP-1`**, etc. will be used. For example, notation **`IP-1`** will denote the first modification of component "INITIAL POPULATION".

*A. Randomly generated population* (**`IP-1`**)

In the simplest way, the genetic algorithm starts from a pure random population. No additional actions (i.e. improvement of the members of an initial population) are involved.

*B. Increased and improved initial population* (**`IP-2`**)

HGA starts with the increased population of the size *INIT_PS=2PS,* where *PS* – the population size. Only $PS$ best members are left. All the members of the initial population are improved by the local improvement procedure.

*C. Improved initial population* (**`IP-3`**)

HGA starts with the randomly generated population of the size *PS*. All the generated individuals are improved.

### 2.4.2 Component "LOCAL IMPROVEMENT"

*A. Quick tabu search* (**`LI-1`**)

The decreased number of TS iterations $\tau = 25$ and the increased number of generations $N_{gen} = 200$ are used. Other parameter values remain the same (as presented in Table 1).

*B. Prolonged tabu search* (**`LI-2`**)

The increased number of TS iterations $\tau = 100$ and the decreased number generations $N_{gen} = 50$ are used. Other parameter values remain unchanged.

*C. Quick hierarchical iterated tabu search* (**`LI-3`**)

The decreased number of iterations of the hierarchical iterated tabu search $Q_{HIER} = 1\times2\times2\times2\times2\times2\times2 = 128$ and the increased number of generations $N_{gen} = 200$ are used. Other parameter values remain unchanged.

*D. Prolonged hierarchical iterated tabu search* (**`LI-4`**)

The increased number of iterations of the hierarchical iterated tabu search $Q_{HIER} = 1\times2\times2\times2\times2\times2\times4 = 512$ and the decreased number of generations $N_{gen} = 50$ are used. Other parameter values remain the same.

*E. Neutral variant* (**`LI-5`**)

All parameter values remain unchanged.

### 2.4.3 Component "SELECTION"

*A. Roulette wheel selection* (**`S-1`**)

In the case of roulette wheel selection [10-12], the scaled fitness values are utilized instead of the pure values of the objective function. The rule is based on the roulette wheel criterion, which to chromosome $i$ in the population of $PS$ chromosomes assigns a selection probability $Pr_i$ proportional to the fitness value as in this equation:

$$Pr_i = \frac{F_i}{\sum_{j=1}^{PS} F_j}; \tag{20}$$

where $F_i$, $F_j$ are scaled fitness values.

*B. Rank-based selection* (**`S-2`**)

Assume that all the members of the current population are sorted in the ascending order of their fitness. According to the rank-based rule [13], the positions $u$ of the parent within the sorted population $P$ are determined according to the formula: $u = \lfloor v^{\sigma} \rfloor$; where $v$ is a uniform random number from the interval $[1, PS^{1/\sigma}]$, where $PS$ is the population size, and $\sigma$ is a real number from the interval $[1, 2]$. We used $\sigma = 2.0$. It is obvious that the better the individual, the larger probability of it being selected for the crossover.

*C. Random selection* (`S-3`)

The parents' chromosomes are chosen in a random way (the fitness of the individuals is not considered).

### 2.4.4 Component "CROSSOVER"

*A. Frequency (backbone)-based greedy crossover* (`C-1`)

This crossover procedure is similar to that described in Section 2.3.3. The difference is that instead of two offspring only one is generated (the generation of the opposition-based offspring is not performed).

*B. Greedy opposition-based crossover* (`C-2`)

The greedy opposition-based crossover, which takes into account the value of the objective function, is used. Two offspring $p^{\circ}$, $p^{\circ\circ}$ are generated.

*C. Greedy crossover* (`C-3`)

The greedy crossover is used and one offspring $p^{\circ}$ is generated.

*D. Heuristic opposition-based crossover* (`C-4`).

The process of heuristic opposition-based crossover takes into account the number of common parent genes. They are the basis for the formation of the offspring. If there are many common genes, "foreign" genes from other individuals are included. The heuristic crossover also performs partial improvement of the offspring.

*E. Heuristic crossover* (`C-5`)

This crossover procedure is similar to `C-4`. Only one offspring is generated instead of two.

*F. Multi-parent frequency and opposition-based crossover* (`C-6`)

Multi-parent frequency and opposition-based crossover uses all the members of the current population to create two offspring [14].

*G. Multi-parent frequency-based crossover* (`C-7`)

This crossover procedure is similar to `C-6`. A single offspring is generated instead of two.

*H. Tailored (problem-specific) opposition-based crossover* (`C-8`)

This crossover is based on calculating the average integer value between the two corresponding values in the parental solutions. Two offspring $p^{\circ}$, $p^{\circ\circ}$ are generated.

*I. Tailored (problem-specific) crossover* (`C-9`)

This crossover procedure is similar to `C-8`. Only one offspring is generated.

*J. Frequency (backbone) and opposition-based crossover* (`C-10`)

This crossover procedure is the same as described in Section 2.3.3.

### 2.4.5 Component "NUMBER OF CROSSOVERS PER GENERATION"

*A. Increased number of crossovers* (`NOC-1`)

In this case, the number of crossovers per generation is equal to $\lambda$ ($\lambda = PS$). The newly produced offspring (juveniles) are added to a pool (temporary population) consisting of $\lambda$ offspring. So, the extended population of $PS + \lambda$ members is maintained. The members of this population are not altered during the current generation. At the end of the generation, $PS$ individuals are selected out of $PS + \lambda$ individuals to form the new population for the next generation.

*B. Single crossover* (`NOC-2`)

One crossover was used per generation.

### 2.4.6 Component "POPULATION REPLACEMENT"

*A. Worst individual replacement* 1 (`PR-1`)

In this scheme, the produced offspring takes the place of the worst member in the current population, but under the necessary condition that the newly produced offspring solution is better than the worst individual.

*B. Worst individual replacement* 2 (`PR-2`)

In this case, the produced offspring replaces the worst individual of the current population ignoring the fitness of this individual.

*C. Worse parent replacement* 1 (`PR-3`)

The current option is very similar to the option `PR-1`. The only difference is that the offspring replaces its worse parent only when the offspring is better than its related parent.

*D. Worse parent replacement 2* (`PR-4`)

This option is very similar to the option `PR-3`, except that the offspring replaces its worse parent disregarding the fitness.

*E. Modified replacement* (`PR-5`)

This is similar to the option `PR-1`. Additionally, it is tested if the offspring is better than the best individual of the current population. If this is the case, then specifically the best individual (instead of the worst individual) is replaced.

### 2.4.7 Component "POPULATION RESTART (RESET)"

*A. No restarts* (`PRS-1`)

In this case, no restarts are involved at all.

*B. Multi-mutation* (`PRS-2`)

The restart from an entirely new population may seem too aggressive. The alternative option is multi-mutation, where the mutation procedure is applied to all the members of the population. The advantage of multi-mutation is that the rate of mutation can be flexibly controlled by the user. In our implementation, the mutation rate is equal to $0.1 \times m$.

*C. Opposition-based reconstruction* (**PRS-3**)

In this variant, the newly constructed solutions are opposition-based solutions with respect to the existing solutions of the current population. The process of generating the opposite solutions is analogous to that used in opposition-based crossover operators.

*D. Gene translocation* (**PRS-4**)

This process is similar to the multi-parent crossover procedure, except that many children are produced instead of a single child.

*E. Chaotic reconstruction* (**PRS-5**)

The population is reconstructed by using the chaotic logistic mapping function (logistic map) (see [15] for details):

$$x_{j+1} = k \times x_j \times (1 - x_j); \tag{21}$$

where $k = 4$, $x_0$ is a real number from the interval $[0, 1]$, $j = 0,1,2, \dots, n - 1$.

*F. Chaotic modified reconstruction* (**PRS-6**)

This option is very similar to **PRS-5**, except that the generated chaotic real numbers are "directly" associated with the integer values of solution elements.

*G. Random reconstruction* (**PRS-7**)

In this case, the generation of a purely random population is used.

The following collection (set) of options—**IP-3**, **LI-5**, **S-3**, **C-10**, **NOC-2**, **PR-5**, **PRS-7**—as the "basic configuration" of our hybrid genetic algorithm. The choice of these options is based on a preliminary experimentation described in [5]. The basic configuration is shortly denoted by **HGA-BV**.

## 3. RESULTS OF COMPUTATIONAL EXPERIMENTS

The algorithm has been tested on the medium and large-scaled GPP instances with $n = 256$ and $n = 1024$. The instances are generated according to the method described in (1)[1]. The grids are of dimensions $16 \times 16$ ($n_1 = n_2 = 16$) and $32 \times 32$ ($n_1 = n_2 = 32$), respectively. The grey density parameter $m$ varies from 2 to 128 and from 2 to 512.

The values of the control parameters of HGA used in the experiments are shown in Table 2.

First, we have experimented with the problems in size 256 and compared our algorithm with the improved genetic-evolutionary algorithm (IGEA) presented in [16]. To our knowledge, IGEA seems very likely to be the most efficient (to date) heuristic algorithm for the problems of this size. As the

[1] These instances can also be found on the website: http://www.personalas.ktu.lt/~alfmise/.

algorithms IGEA and HGA constantly find the best known (pseudo-optimal) solutions (BKSs), the run time performance, rather than the quality of solutions is compared.

The results demonstrate that HGA clearly dominates IGEA (see Table 3). IGEA was able to slightly outperform HGA only in a very few cases ($m = 26, 101, 102, 103$). Figure 4 illustrates overall speed improvement of HGA ($m$'s varies from 30 to 100).

During an additional extensive, long-lasting experimentation, the algorithm HGA on the large-sized problems ($n = 1024$), which are much more difficult and time-consuming, has been examined. It should be pointed out that we have succeeded in discovering new record-breaking solutions for more than 190 values of $m$. The deviations of the new best known solutions found from the previous best solutions are shown in Figure 3.

## 3.1 Results of comparison of algorithms

**Table 2.** Values of the control parameters of the hybrid genetic algorithm used in the comparison

| Parameter | Value | Remarks |
|---|---|---|
| Population size, $PS$ | 20 | |
| Number of generations, $N_{gen}$ | 40 | |
| Idle generations limit, $L_{idle\_gen}$ | $\lfloor 0.15 N_{gen} \rfloor$ | $0 < L_{idle\_gen} \le N_{gen}$ |
| Distance threshold, $DT$ | $\lfloor 0.25m \rfloor$ | $0 \le DT \le m$ |
| Number of hierarchical tabu search iterations, $Q_{HIER}$ | 384 | $Q_{HIER} = Q \times Q_1 \times Q_2 \times Q_3 \times Q_4 \times Q_5 \times Q_6 \times Q_7$ [†] |
| Number of tabu search iterations, $\tau$ | 80 | |
| Tabu tenure, $h$ | $\lfloor 0.3m \rfloor$ | $h > 0$ |
| Idle iterations limit, $L_{idle\_iter}$ | $\lfloor 0.2\tau \rfloor$ | $0 < L_{idle\_iter} \le \tau$ |
| Neighbourhood size factor, $\rho$ | 0.4 | $\rho > 0$ |
| Randomization coefficient, $\alpha$ | 0.02 | $0 < \alpha < 1$ |
| Mutation rate, $\mu$ | $\lfloor 0.15m \rfloor$ | $0 < \mu < m$ |

[†] $Q = Q_1 = Q_2 = Q_3 = Q_4 = Q_5 = Q_6 = 2$, $Q_7 = 3$.

**Table 3.** Results of the experiments with the medium-sized GPP instances (*n* = 256)

| *m* | Best known value (BKV) | Dev. from BKV | CPU time (sec) |  | *m* | Best known value (BKV) | Dev. from BKV | CPU time (sec) |  | *m* | Best known value (BKV) | Dev. from BKV | CPU time (sec) |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  |  |  | IGEA | NHGA |  |  |  | IGEA | NHGA |  |  |  | IGEA | NHGA |
| **2** | 1562 | 0 | ***0.0*** | ***0.0045*** |  | 8674910 | 0 | *150.0* | ***5.91*** | **87** | 39389054 | 0 | *25.0* | ***0.19*** |
| **3** | 7810 | 0 | ***0.0*** | ***0.0046*** |  | 9129192 | 0 | *64.0* | ***11.38*** | **88** | 40416536 | 0 | *23.0* | ***0.15*** |
| **4** | 15620 | 0 | ***0.0*** | ***0.0047*** |  | 9575736 | 0 | *3.1* | ***0.29*** | **89** | 41512742 | 0 | *183.0* | ***6.98*** |
| **5** | 38072 | 0 | ***0.0*** | ***0.0048*** |  | 10016256 | 0 | *2.0* | ***0.16*** | **90** | 42597626 | 0 | *165.0* | ***4.76*** |
| **6** | 63508 | 0 | ***0.0*** | ***0.0049*** |  | 10518838 | 0 | *3.4* | ***0.56*** | **91** | 43676474 | 0 | *224.0* | ***17.64*** |
| **7** | 97178 | 0 | ***0.0*** | ***0.0050*** |  | 11017342 | 0 | *2.8* | ***0.56*** | **92** | 44759294 | 0 | *157.0* | ***7.94*** |
| **8** | 131240 | 0 | ***0.0*** | ***0.0051*** |  | 11516840 | 0 | *7.5* | ***0.83*** | **93** | 45870244 | 0 | *214.0* | ***19.18*** |
| **9** | 183744 | 0 | ***0.0*** | ***0.0052*** |  | 12018388 | 0 | *6.3* | ***0.39*** | **94** | 46975856 | 0 | *190.0* | ***21.52*** |
| **10** | 242266 | 0 | ***0.0*** | ***0.0053*** |  | 12558226 | 0 | *4.6* | ***0.54*** | **95** | 48081112 | 0 | *169.0* | ***2.14*** |
| **11** | 304722 | 0 | *0.1* | ***0.0054*** |  | 13096646 | 0 | *4.0* | ***0.03*** | **96** | 49182368 | 0 | *216.0* | ***6.58*** |
| **12** | 368952 | 0 | *0.1* | ***0.0055*** |  | 13661614 | 0 | *10.1* | ***0.12*** | **97** | 50344050 | 0 | *213.0* | ***23.88*** |
| **13** | 457504 | 0 | *0.1* | ***0.0056*** |  | 14229492 | 0 | *2.8* | ***0.15*** | **98** | 51486642 | 0 | *188.0* | ***63.40*** |
| **14** | 547522 | 0 | *0.1* | ***0.0057*** |  | 14793682 | 0 | *2.2* | ***0.37*** | **99** | 52660116 | 0 | *201.0* | ***50.53*** |
| **15** | 644036 | 0 | *0.1* | ***0.0058*** |  | 15363628 | 0 | *2.3* | ***0.09*** | **100** | 53838088 | 0 | *117.0* | ***48.18*** |
| **16** | 742480 | 0 | *0.1* | ***0.0059*** |  | 15981086 | 0 | *3.5* | ***0.71*** | **101** | 55014262 | 0 | ***84.0*** | *156.00* |
| **17** | 878888 | 0 | *0.2* | ***0.0060*** |  | 16575644 | 0 | *2.4* | ***0.95*** | **102** | 56202826 | 0 | ***40.0*** | *115.00* |
| **18** | 1012990 | 0 | *0.1* | ***0.0061*** |  | 17194812 | 0 | *2.2* | ***0.01*** | **103** | 57417112 | 0 | ***73.0*** | *84.00* |
| **19** | 1157992 | 0 | *0.2* | ***0.0062*** |  | 17822806 | 0 | *3.6* | ***0.01*** | **104** | 58625240 | 0 | *62.0* | ***51.14*** |
| **20** | 1305744 | 0 | *0.3* | ***0.0863*** |  | 18435790 | 0 | *1.9* | ***0.00*** | **105** | 59854744 | 0 | *38.0* | ***32.41*** |
| **21** | 1466210 | 0 | *0.5* | ***0.0064*** |  | 19050432 | 0 | *2.3* | ***0.00*** | **106** | 61084902 | 0 | *33.0* | ***10.85*** |
| **22** | 1637794 | 0 | *0.3* | ***0.0065*** |  | 19848790 | 0 | *3.1* | ***0.00*** | **107** | 62324634 | 0 | *21.0* | ***0.73*** |
| **23** | 1820052 | 0 | *0.2* | ***0.0066*** |  | 20648754 | 0 | *4.5* | ***0.02*** | **108** | 63582416 | 0 | *12.6* | ***0.65*** |
| **24** | 2010846 | 0 | *0.6* | ***0.0267*** |  | 21439396 | 0 | *9.7* | ***0.08*** | **109** | 64851966 | 0 | *11.1* | ***1.02*** |
| **25** | 2215714 | 0 | *3.2* | ***0.3168*** |  | 22234020 | 0 | *18.0* | ***0.23*** | **110** | 66120434 | 0 | *10.7* | ***0.41*** |
| **26** | 2426298 | 0 | ***16.5*** | *22.98*69 |  | 23049732 | 0 | *27.0* | ***0.64*** | **111** | 67392724 | 0 | *8.2* | ***0.46*** |
| **27** | 2645436 | 0 | *1.1* | ***0.0270*** |  | 23852796 | 0 | *26.0* | ***0.98*** | **112** | 68666416 | 0 | *7.7* | ***0.17*** |
| **28** | 2871704 | 0 | *0.9* | ***0.0371*** |  | 24693608 | 0 | *78.0* | ***0.43*** | **113** | 69984758 | 0 | *10.2* | ***0.13*** |
| **29** | 3122510 | 0 | *0.7* | ***0.0372*** |  | 25522408 | 0 | *490.0* | ***64.80*** | **114** | 71304194 | 0 | *6.3* | ***0.18*** |
| **30** | 3373854 | 0 | *0.5* | ***0.0073*** |  | 26375828 | 0 | *298.0* | ***5.81*** | **115** | 72630764 | 0 | *5.1* | ***0.37*** |
| **31** | 3646344 | 0 | *0.6* | ***0.0074*** |  | 27235240 | 0 | *304.0* | ***2.50*** | **116** | 73962220 | 0 | *5.3* | ***0.21*** |
| **32** | 3899744 | 0 | *0.5* | ***0.0275*** |  | 28114952 | 0 | *41.0* | ***0.85*** | **117** | 75307424 | 0 | *4.0* | ***0.03*** |
| **33** | 4230950 | 0 | *0.7* | ***0.0376*** |  | 29000908 | 0 | *121.0* | ***1.30*** | **118** | 76657014 | 0 | *3.6* | ***0.06*** |
| **34** | 4560162 | 0 | *2.6* | ***0.3677*** |  | 29894452 | 0 | *145.0* | ***4.81*** | **119** | 78015914 | 0 | *2.3* | ***0.03*** |
| **35** | 4890132 | 0 | *3.2* | ***0.4178*** |  | 30797954 | 0 | *117.0* | ***1.15*** | **120** | 79375832 | 0 | *1.7* | ***0.05*** |
| **36** | 5222296 | 0 | *2.0* | ***0.4479*** |  | 31702182 | 0 | *11.6* | ***0.81*** | **121** | 80756852 | 0 | *1.6* | ***0.07*** |
| **37** | 5565236 | 0 | *1.8* | ***0.3480*** |  | 32593088 | 0 | *3.3* | ***0.47*** | **122** | 82138768 | 0 | *1.4* | ***0.03*** |
| **38** | 5909202 | 0 | *0.9* | ***0.1481*** |  | 33544628 | 0 | *3.9* | ***0.26*** | **123** | 83528554 | 0 | *1.0* | ***0.04*** |
| **39** | 6262248 | 0 | *1.1* | ***0.0882*** |  | 34492592 | 0 | *70.0* | ***3.33*** | **124** | 84920540 | 0 | *0.7* | ***0.01*** |
| **40** | 6613472 | 0 | *0.9* | ***0.0283*** |  | 35443938 | 0 | *57.0* | ***1.41*** | **125** | 86327812 | 0 | *0.4* | ***0.00*** |
| **41** | 7002794 | 0 | *0.6* | ***0.1184*** |  | 36395172 | 0 | *61.0* | ***2.77*** | **126** | 87736646 | 0 | *0.3* | ***0.00*** |
| **42** | 7390586 | 0 | *0.7* | ***0.1685*** |  | 37378800 | 0 | *151.0* | ***1.20*** | **127** | 89150166 | 0 | *0.2* | ***0.00*** |
| **43** | 7794422 | 0 | *3.2* | ***0.2086*** |  | 38376438 | 0 | *94.0* | ***0.32*** | **128** | 90565248 | 0 | *0.2* | ***0.00*** |
| **44** | 8217264 | 0 | *16.0* | ***0.87*** |  |  |  |  |  |  |  |  |  |  |

**Note.** The deviation from BKV (Dev. from BKV) is calculated as the ratio $\left(z^{*} - BKV\right)/BKV$, where $z^{*}$ denotes the algorithms' best-found solution. The best known values of the objective function corresponding to the best known solutions are from [16].

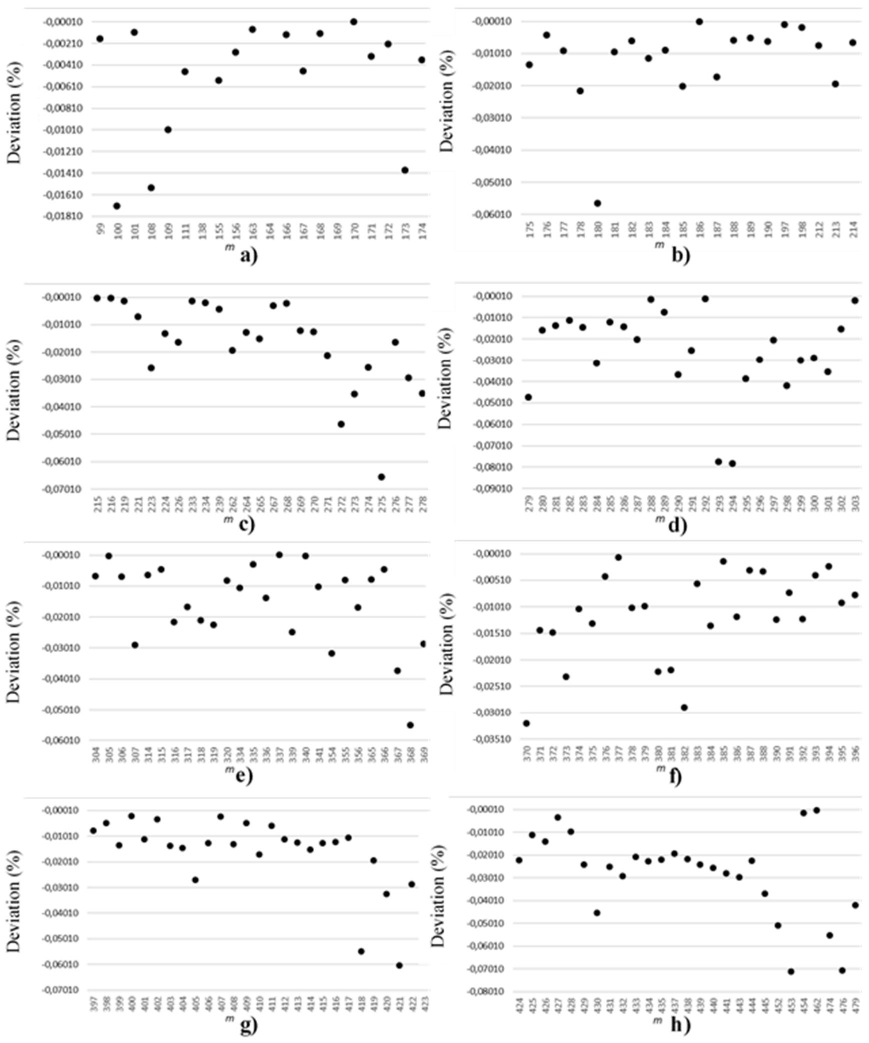

**Figure 3.** The deviations of the new best known solutions found ($n = 1024$): a) $m's$ varies from 99 to 174; b) $m's$ varies from 175 to 214; c) $m's$ varies from 215 to 278; d) $m's$ varies from 279 to 303; e) $m's$ varies from 304 to 369; f) $m's$ varies from 370 to 396; g) $m's$ varies from 397 to 423; h) $m's$ varies from 424 to 479.

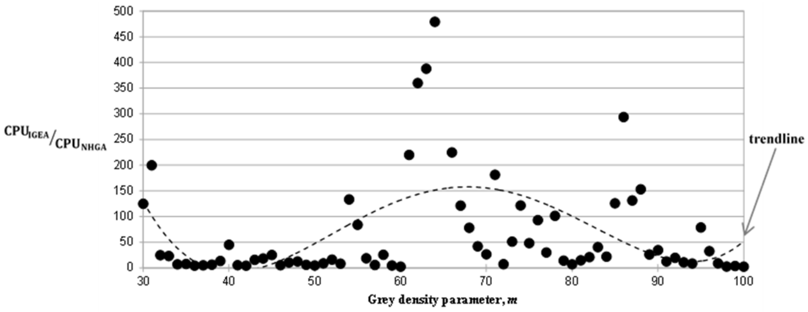

**Figure 4.** Illustration of the run time improvement

Figures 5 and 6 provide some graphical representations (grey frames) corresponding to $m = 401, 402, 403, 404, 405, 406, 407, 408$.

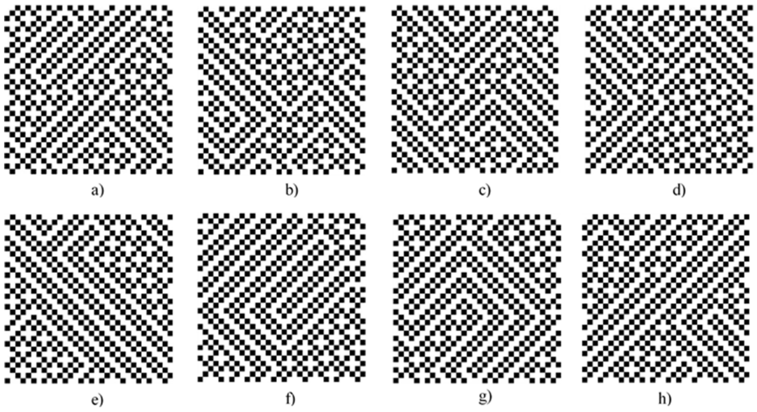

**Figure 5.** Examples of (pseudo-)optimal grey frames ($32 \times 32$, $n = 1024$): (a) $m = 401$, (b) $m = 402$, (c) $m = 403$, (d) $m = 404$, (e) $m = 405$, (f) $m = 406$, (g) $m = 407$, (h) $m = 408$

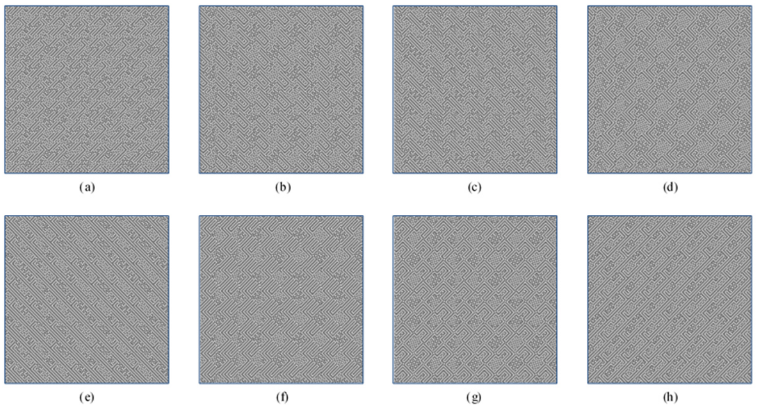

**Figure 6.** Examples of (pseudo-)optimal grey frames[2] ($n = 1024$): (a) $m = 401$, (b) $m = 402$, (c) $m = 403$, (d) $m = 404$, (e) $m = 405$, (f) $m = 406$, (g) $m = 407$, (h) $m = 408$

## 3.2 Results of the experiments with algorithm components

The algorithm has been tested in the medium and large-scaled GPP instances with $n = 256$ and $n = 1024$, respectively. The values of the grey density parameter $m$ vary from $95$ to $104$ for $n = 256$. For $n = 1024$, the values of $m$ are as follows: $50, 60, 70, 80, 90, 100, 110, 120, 130, 140$.

As a performance criterion for our algorithm, the average relative percentage deviation ($\bar{\theta}$) of the yielded solutions from the best known solution (BKS) is used. It is calculated according to the following formula: $\bar{\theta} = 100(\bar{z} - BKV)/BKV$ [%], where $\bar{z}$ is the average objective function value over $10$ runs of the algorithm, while $BKV$ denotes the best known value of the objective function that corresponds to BKS. At every run, the algorithm is applied to the given values of $n$ and $m$, each time starting from new random initial populations. Note that the current run is interrupted if BKS is found, even without reaching the maximum number of generations, $N_{gen}$.

The average deviations of the examined components are presented in Figures 7, 8.

[2] In the graphical illustrations, each $1024$-square-grid is replicated 8 times horizontally and 8 times vertically for a better visibility.

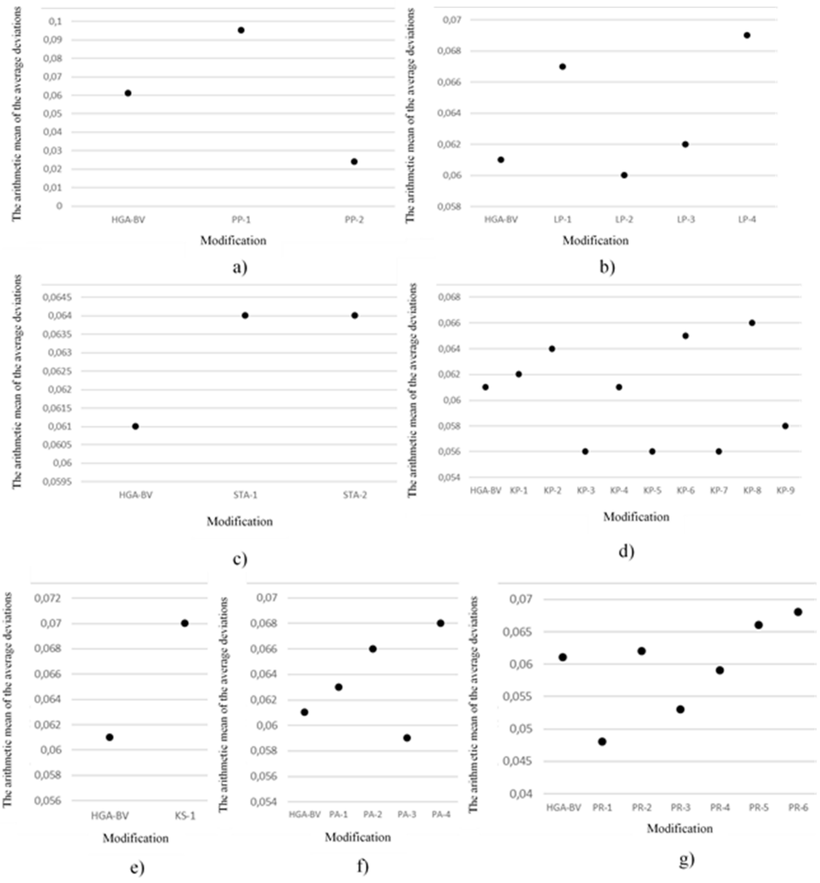

**Figure 7.** The average deviations of the examined components ($m = 95, 96, \ldots, 104$, $n = 254$): a) "INITIAL POPULATION"; b) "LOCAL IMPROVMENT"; c) "SELECTION", d) "CROSSOVER"; e) "NUMBER OF CROSSOVERS PER GENERATION"; f) "POPULATION REPLACEMENT"; g) "POPULATION RESTART (RESET)"

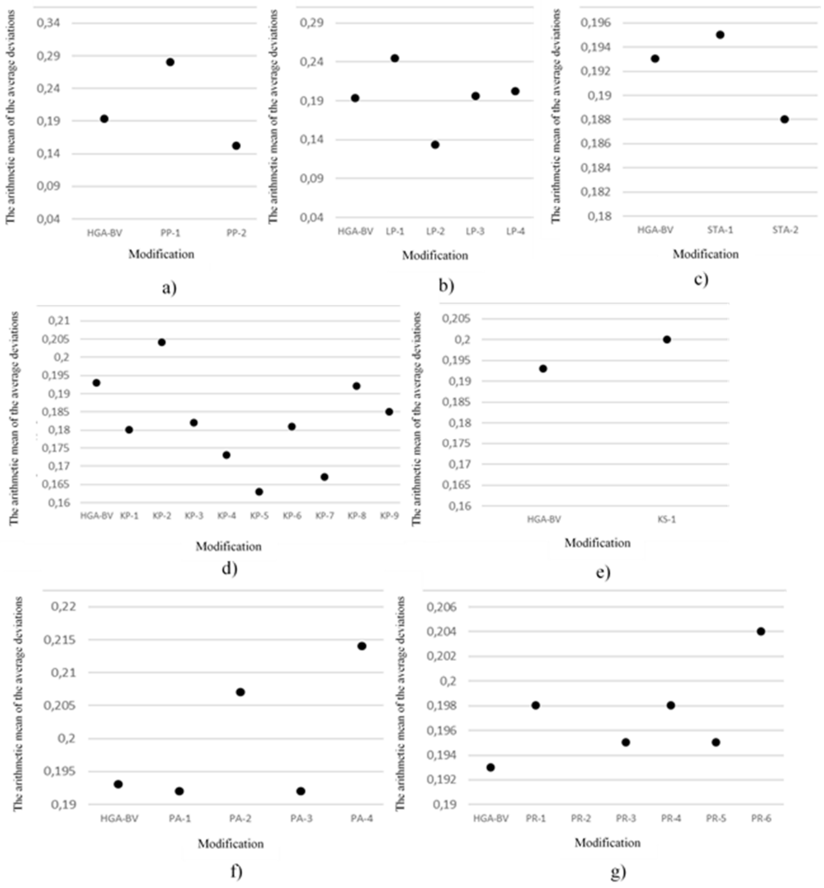

**Figure 8.** The average deviations of the examined components ($m = 50, 60, \dots, 140$, $n = 1024$): a) "INITIAL POPULATION"; b) "LOCAL IMPROVMENT"; c) "SELECTION", d) "CROSSOVER"; e) "NUMBER OF CROSSOVERS PER GENERATION"; f) "POPULATION REPLACEMENT"; g) "POPULATION RESTART (RESET)"

## CONCLUSIONS

1. A comprehensive theoretical review has shown that the use of hybrid algorithms in solving difficult combinatorial tasks is an effective way to find high quality (pseudo-optimal) solutions within an acceptable execution time.
2. A new improved hybrid genetic algorithm for solving the grey pattern quadratic assignment problem has been proposed. A compacted, reduced

neighbourhood is used. This enables very fast execution of the hierarchical ITS algorithm and the HGA. Moreover, a smart combination of the iterated tabu search and the greedy adaptive perturbations is applied. This allows for a beneficial balance between diversification and intensification during the iterative search process.

3. The experimental evaluation of HGA efficiency suggests that HGA is more efficient in terms of CPU time (when experiments are performed with a medium-sized task, $n = 256$) and in terms of the quality of solutions (when experiments are performed with a large-sized task, $n = 1024$).

   The results of the computational experiments with different components demonstrate:
   - using increased, improved initial populations is indeed much more advantageous than the use of random populations;
   - better results are obtained by increasing the number of tabu search iterations;
   - the rank-based selection procedure is only slightly better than the remaining procedures;
   - better results are obtained by the frequency(backbone)-based greedy crossover, the greedy crossover and the heuristic crossover. Generation of opposition offspring has no significant effect on the efficiency of the algorithm;
   - it is more efficient when immediate population renewal is used (i.e. one crossover per generation is applied);
   - it is recommended to use the option, where the offspring replaces the worse parent;
   - multi-mutation based population restarts are an effective way to avoid stagnation in the search process.

## LIST OF REFERENCES

1. ÇELA, E. *The Quadratic Assignment Problem: Theory and Algorithms*. Dordrecht: Kluwer, 1998.

2. TAILLARD, E. Comparison of iterative searches for the quadratic assignment problem. *Location Science*. 1995, Vol. 3, 87–105.

3. STANEVIČIENĖ, E., MISEVIČIUS, A. Empirical Study of Local Optimization Methods. *ALTA'17: Pažangios mokymosi technologijos: išmanusis mokymasis: tarptautinė konferencija skirta IT idėjų sklaidai: konferencijos pranešimų medžiaga*. Kaunas: Kauno technologijos universitetas, 2017, 91–99.

4. ZHOU, Y., HAO, J.-K., DUVAL, B. Opposition-based memetic search for the maximum diversity problem. *IEEE Transactions on Evolutionary Computation*. 2017, Vol. 21, 731−745.

5. MISEVIČIUS, A., STANEVIČIENĖ, E. A new hybrid genetic algorithm for the grey pattern quadratic assignment problem. *Information Technology and Control*. 2018, Vol. 47(3), 503–520.

6. HUSSIN, M.S., STÜTZLE, T. Hierarchical iterated local search for the quadratic assignment problem. In M.J. Blesa, C. Blum, L. Di Gaspero, A. Roli, M. Sampels, A. Schaerf (eds.). *Hybrid Metaheuristics. HM 2009. Lecture Notes in Computer Science*. Berlin, Heidelberg: Springer, 2009, Vol. 5818, 115−129.

7. WU, Q., HAO, J.-K. A hybrid metaheuristic method for the maximum diversity problem. *European Journal of Operational Research*. 2013, Vol. 231, 452–464.

8. DELL'AMICO, M., TRUBIAN, M. Solution of large weighted equicut problems. *European Journal of Operational Research*. 1998, Vol. 106, 500–521.

9. STANEVIČIENĖ, E., MISEVIČIUS, A., OSTREIKA, A. Experimental Analysis of Hybrid Genetic Algorithm for the Grey Pattern Quadratic Assignment Problem. *Information Technology and Control.* 2019, Vol. 48(2), 335–356.

10. GOLDBERG, D.E. *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading: Addison-Wesley, 1989.

11. RAZALI, N., M., JOHN, G. Genetic Algorithm Performance with Different Selection Strategies in Solving TSP. *In: The 2011 International Conference of Computational Intelligence and Intelligent Systems*. London: Imperial College, 2011, Vol. 2, 4–9.

12. ZHONG, J., HU, X., ZHANG, J., GU, M. Comparison of Performance between Different Selection Strategies on Simple Genetic Algorithms. *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)*. Vienna, 2005, 1115–1121.

13. TATE, D.M., SMITH, A.E. A genetic approach to the quadratic assignment problem. *Computers & Operations Research*. 1995, Vol. 1, 73–83.

14. EIBEN, A.E., RAUE, P.-E., RUTTKAY, Z. Genetic algorithms with multi-parent recombination. In Y. Davidor, H.P. Schwefel, R. Männer (eds.). *Parallel Problem Solving from Nature – PPSN III, International Conference on Evolutionary Computation, The Third Conference on Parallel Problem Solving from Nature, Proceedings. Lecture Notes in Computer Science*. Berlin-Heidelber: Springer, 1994, Vol. 866, 78-87.

15. CAPONETTO, R., FORTUNA, L., FAZZINO, S., XIBILIA, M.G. Chaotic sequences to improve the performance of evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*. 2003, Vol. 7(3), 289–304.

16. MISEVIČIUS, A. Generation of grey patterns using an improved genetic-evolutionary algorithm: some new results. *Information Technology and Control*. 2011, Vol. 40, 330–343.

## LIST OF PUBLICATIONS ON THE SUBJECT OF DISSERTATION

**Articles Published in Journals Belonging to Scientific International Databases (Indexed in the Web of Science with Impact Factor):**

1. MISEVIČIUS, A., STANEVIČIENĖ, E. A New Hybrid Genetic Algorithm for the Grey Pattern Quadratic Assignment Problem. *Information Technology and Control.* 2018, Vol. 47(3), 503–520.
2. STANEVIČIENĖ, E., MISEVIČIUS, A., OSTREIKA, A. Experimental Analysis of Hybrid Genetic Algorithm for the Grey Pattern Quadratic Assignment Problem. *Information Technology and Control.* 2019, Vol. 48(2), 335–356.

**Publications in Other International Databases:**

1. MISEVIČIUS, A., BLONSKIS, J., BUKŠNAITIS, V., STANEVIČIENĖ, E., ŽELVYS, T. Aukštos kokybės skaitmeninių spalvų atspalvių kompiuterinis generavimas. *Computational Science and Techniques.* Klaipėda: Klaipėda University, 2013, Vol. 1, no. 2, 126–140.
2. MISEVIČIUS, A., GUOGIS, E., STANEVIČIENĖ, E. Computational algorithmic generation of high-quality colour patterns. *Information and software technologies: 19th international conference, ICIST 2013, Kaunas, Lithuania, October 10-11, 2013: proceedings / [edited by] Tomas Skersys, Rimantas Butleris, Rita Butkiene. Communications in computer and information science.* Berlin, Heidelberg: Springer, 2013, Vol. 403, 285–296.
3. STANEVIČIENĖ, E., MISEVIČIUS, A., KUZNECOVAITĖ, D., DRĄSUTĖ, V. Hybrid genetic algorithms for image processing: a short review. *ALTA'15: Pažangios mokymosi technologijos: konferencijos*

*pranešimų medžiaga*. Kaunas: Kauno technologijos universitetas, 2015, 45–49.

4. STANEVIČIENĖ, E., MISEVIČIUS, A. Empirical Study of Local Optimization Methods. *ALTA'17: Pažangios mokymosi technologijos: išmanusis mokymasis: tarptautinė konferencija skirta IT idėjų sklaidai: konferencijos pranešimų medžiaga*. Kaunas: Kauno technologijos universitetas, 2017, 91–99.

## INFORMATION ABOUT THE AUTHOR OF THE DISSERTATION

**Education:**
2002–2009: Bachelor's degree in Economics at Kaunas University of Technology, School of Economics and Business.
2010–2012: Master's degree in Information technology at Kaunas University of Technology, Faculty of Informatics.
2013–2018: doctoral studies in Informatics at Kaunas University of Technology, Faculty of Informatics.
**Work experience:**
Since 2016: Lecturer at Kaunas University of Technology, Faculty of Informatics.
**E-mail:**
evelina.staneviciene@ktu.lt

## REZIUMĖ

Gana aktuali informatikos mokslo sritis yra optimizavimo metodai ir algoritmai, jų sudarymas ir tyrimas. Reikšmingą šių metodų grupę sudaro modernieji euristiniai algoritmai, tame tarpe ir hibridiniai genetiniai algoritmai.

Modernieji euristiniai algoritmai sėkmingai išvystyti, sprendžiant sudėtingus teorinius-matematinius ir taikomuosius-praktinius uždavinius. Tokio uždavinio pavyzdys yra kvadratinio paskirstymo uždavinys (angl. *quadratic assignment problem*) ir jo atskiras atvejis – pilkų šablonų formavimo uždavinys (angl. *grey pattern problem*).

Šiame darbe koncentruojamasi būtent į efektyvių euristinių algoritmų kūrimą ir taikymą pilkų šablonų formavimo uždaviniui spręsti, kuomet siekiama gauti kuo tikslesnį (optimalų) pilkąjį, iš anksto apibrėžto intensyvumo, atspalvį. Darbe siekta tiek teorinių - iš pradžių buvo sukurti efektyvių euristinių algoritmų prototipai, skirti sudėtingai teorinių kombinatorinių optimizavimo uždavinių klasei, tiek praktinių tikslų – kuomet prototipai adaptuoti konkrečiam praktiniam-techniniam uždaviniui.

Duotam uždaviniui gali būti pritaikyti įvairių tipų euristiniai algoritmai, pradedant lokalios paieškos algoritmais, baigiant biologinių procesų inspiruotais kolektyvinio intelekto imitavimo algoritmais. Svarbus aspektas yra susijęs su hibridizavimu, kuomet bandoma pasiekti naudingą sąveiką kombinuojant (derinant) skirtingus komponentus (algoritmus). Geras pavyzdys galėtų būti greitos vieno sprendinio pagrindu paremtos euristikos ir populiacija paremto (evoliucinio) algoritmo kombinavimas. Kitas svarbus aspektas yra prieš tai sukonstruotų efektyvių euristinių algoritmų daugkartinis panaudojimas.

**Tyrimo objektas.** Tyrimo objektas yra hibridiniai genetiniai algoritmai, skirti pilkų šablonų formavimo uždaviniui spręsti.

**Tyrimo tikslas.** Tyrimo tikslas yra ištirti hibridinio genetinio algoritmo modifikacijų pilkų šablonų formavimo uždaviniui veikimo efektyvumą ir nustatyti efektyviausius algoritmo variantus.

**Tyrimo uždaviniai.** Tikslui pasiekti buvo iškelti tokie darbo uždaviniai:

1. Esamų algoritminių sprendimo būdų pilkų šablonų formavimo uždaviniui žvalgomasis tyrimas.
2. Naujo hibridinio genetinio algoritmo pilkų šablonų formavimo uždaviniui pasiūlymas ir realizavimas.
3. Algoritmo efektyvumo eksperimentinis ištyrimas ir gautų rezultatų analizė.

**Tyrimo metodika.** Tyrimų metodika remiasi euristinių ir metaeuristinių kombinatorinio optimizavimo algoritmų panaudojimu, kompiuterine algoritmų efektyvumo analize.

**Darbo mokslinis naujumas ir praktinė vertė**. Mokslinį naujumą charakterizuoja du aspektai.

1. Pasiūlytas patobulintos struktūros (architektūros) hibridinis genetinis algoritmas pilkų šablonų formavimo uždaviniui spręsti.
2. Realizuotas hibridinio genetinio algoritmo integravimas su naujo tipo iteratyviosios tabu paieškos procedūra, kombinuojama su efektyviu adaptyviu sprendinių pertvarkymu.

Praktinė vertė yra ta, kad sudarytas algoritmas sėkmingai išbandytas, sprendžiant didelės apimties pilkų šablonų formavimo uždavinius. Tai įgalina padidintos kokybės spalvų atspalvių, skaitmeninių vaizdų sukūrimą. Visa tai yra aktualu, pritaikant modernius euristinius algoritmus praplėstoje/virtualioje realybėje.

## IŠVADOS

1. Atlikta išsami literatūros apžvalga parodė, kad hibridinių algoritmų taikymas, sprendžiant sunkius kombinatorinius uždavinius, yra efektyvus būdas, siekiant surasti aukštos kokybės (galimai optimalius) sprendinius, per priimtiną vykdymo laiką.
2. Pasiūlytas ir realizuotas inovatyvus hibridinis genetinis algoritmas (HGA). Esminės HGA savybės: a) algoritme įkomponuota hierarchinė iteratyvioji tabu paieška; b) panaudojama labai aukštos kokybės diversifikuotų sprendinių populiacija; c) pritaikytas efektyvus tabu

paieškos ir godžiojo adaptyviojo sprendinių pertvarkymo kombinavimas.

3. Atliktas HGA efektyvumo eksperimentinis tyrimas leidžia daryti išvadą, kad HGA yra efektyvesnis laiko atžvilgiu (kuomet palyginamieji eksperimentai atlikti su vidutinio dydžio uždaviniu, $n = 256$) ir sprendinių kokybės atžvilgiu (kuomet palyginamieji eksperimentai atlikti su didelio dydžio uždaviniu, $n = 1024$) už iki šiol žinomus algoritmus, skirtus spręsti PŠF uždavinį.

   Atlikti eksperimentai su skirtingomis algoritmo komponentų modifikacijomis įtakojo tinkamo komponentų rinkinio pasirinkimą išplėstiniams eksperimentams. HGA komponentų ištyrimas ir eksperimentinis įvertinimas parodė, kad suderinus komponentų rinkinį iš komponentų modifikacijų galima gauti efektyvesnius rezultatus.
4. Nustatyta, jog efektyviausi yra šie komponentų variantai:
   - pagerinta pradinė populiacija;
   - padidintas sprendinių pagerinimo (t.y., tabu paieškos) iteracijų skaičius;
   - kryžminimo procedūros, įvertinančios tikslo funkciją ir uždavinio specifiką;
   - operatyvus populiacijos atnaujinimas, atsižvelgiant į atstumo kriterijų;
   - nedidelio lygio mutacijomis besiremiančių populiacijos perkrovimų panaudojimas.

UDK 004.421.4(043.3)

SL344. 2019-07-03, 2,25 leidyb. apsk. l. Tiražas 50 egz.
Išleido Kauno technologijos universitetas, K. Donelaičio g. 73, 44249 Kaunas
Spausdino leidyklos „Technologija“ spaustuvė, Studentų g. 54, 51424 Kaunas