Adv Syst Sci Appl 2023; **02**:164–177
Published online at https://ijassa.ipu.ru.

Pairwise Similarity Estimation for Discrete Optimization Problems

Darya Lemtyuzhnikova[1,2*], Pavel Chebotarev[3,4], Mikhail Goubko[5], Nikita Shushko[1], Mikhail Somov[1]

[1] *Institute of Control Sciences of RAS, Moscow, Russia*

[2] *Moscow Aviation Institute, Moscow, Russia*
[3] *Technion – Israel Institute of Technology, Haifa, Israel*

[4] *Kharkevich Institute for Information Transmission Problems of RAS, Moscow, Russia*

[5] *Flöha, Altran Deutschland S.A.S. & Co. KG*

Abstract: In this paper, we propose a new method, called the pairwise similarity method, for assessing the similarity of instances of optimization problems. This method is a generalization of the metric approach proposed earlier for scheduling problems. It relaxes the requirements to the structure of the problem constraints. The method involves a dissimilarity function for the comparison of instances. It identifies "simple" instances that can be solved in polynomial time and uses them to get good approximations for other instances. We apply the pairwise similarity method to two discrete optimization problems: the majority domination problem and the maximum cut problem.

Keywords: pairwise similarity method, pairwise dissimilarity function, discrete optimization, NP-hard problems, special case, majority domination problem, maximum cut problem

1. INTRODUCTION

The problem of NP-hardness is related to one of the key problems of theoretical computer science of the relationship between complexity classes P and NP. To date, there is no way to quantify the complexity of NP-hard problems. Analyzing the behavior of heuristic algorithms as applied to various problem instances leads to the identification of patterns in the data. Studying such patterns allows one to gain insight into the complexity structure of NP-hard problems.

In this paper, we propose a general method for estimating the computational complexity of NP-hard problems based on comparing the problem instances using a dissimilarity function. The proposed method is based on the metric approach, which was originally formulated for scheduling problems [2, 4, 6, 7, 8].

The metric approach was tested for problem instances of the same dimension without taking into account the difference in the structure of the constraints. First, instances were found that enables obtaining an exact solution in a reasonable time. To compare a general instance with such an instance, metrics were proposed that evaluate the difference between the objective functions of those two instances. An exact solution to the first instance induces an approximate solution for the second instance and the characteristics of the latter solution

*Corresponding author: darabbt@gmail.com

have been assessed using the above metrics. More specifically, the metric was defined as the upper bound for the absolute error of the approximate solution. Computational experiments for solving some problems of scheduling theory by approximate algorithms in polynomial time have shown that with increasing dimension, the upper bound for the relative error of the objective function tends to zero, while the proportion of problems that can be solved exactly in polynomial time tends to 100%. This means that the proposed algorithms can be effectively used for most of the instances of the problems under consideration. This made it possible to obtain exact or fairly close to exact solutions. Moreover, with an increase in the dimension of the problem, the proportion of such instances increases [6, 7, 8].

Furthermore, [2] studied the possibility of finding approximate schedules using the metric approach for an NP-complete scheduling problem on two parallel identical machines with priority delays for jobs or for jobs of length 1 or 2 with minimization of execution time. An application of these methods for finding the optimal solution for $P2 \,|\, \text{pred},\ p_j = 1 \,|\, C_{\max}$ using the Coffman and Sethi algorithms has been implemented. In [4], the metric approach was developed for planning a two-station single-track railway. New metrics were proposed and computational tests were carried out.

The pairwise similarity method generalizes the metric approach. It allows for different structures of instances and involves pairwise dissimilarity functions for which satisfaction of the axioms of metric is not required. This paper proposes a general method for assessing heuristics called the pairwise similarity method. It is applied to two discrete optimization problems: the problem of majority domination on graphs with a limited degree of vertices and the maximum cut problem. The focus is on finding instances with the required characteristics and using them to obtain efficient solutions for other instances. Relations are established between the change in the value of the objective function and the difference between the graphs of the problems under consideration.

The structure of the paper is as follows. Section 2 introduces the pairwise similarity method and describes the algorithms that make it up. In Section 3.1, the pairwise similarity method is applied for finding the strict majority domination number in a graph with cycles in the framework of a two-level voting model. In particular, it is proved that graphs with a limited cyclomatic number form a "semisimple" special case, a pairwise dissimilarity function is introduced for them, and an approximate algorithm of polynomial complexity is constructed that has a guaranteed bound for the absolute error of the objective function. In Section 3.2, the pairwise similarity method is applied to the maximum cut problem in the case where it is difficult to analytically obtain an absolute upper bound for the error and the estimates are based on numerical simulation. In conclusion, a research program for further study of the proposed method is outlined.

2. PAIRWISE SIMILARITY METHOD

We first introduce the formal definitions needed to construct the pairwise similarity method.

Definition 2.1. For an NP-hard problem P and a pattern α, the *special case* P^α is the subclass of P comprising all instances $A \in P^\alpha$ that satisfy α.

Definition 2.2. *A simple special case* P^α_{simple} is a special case for which there is a polynomial (or pseudopolynomial) algorithm that finds the exact solution for all instances $A \in P^\alpha_{\text{simple}}$.

Definition 2.3. *A semisimple special case* $P^\alpha_{\text{ssimple}}$ is a special case for which there is a polynomial (or pseudopolynomial) algorithm that finds an approximate solution with a guaranteed error for all instances $A \in P^\alpha_{\text{ssimple}}$.

Definition 2.4. *A numerical special case* P^α_{num} is a special case for which there exist a polynomial (or pseudopolynomial) algorithm and an error estimate such that this estimate for the algorithm is not proved analytically, but is confirmed by numerical experiments on P^α_{num}.

Definition 2.5. *A hard special case* P^{α}_{hard} is a special case for which there is no polynomial (or pseudopolynomial) algorithm that finds exact solutions for all instances $A \in P^{\alpha}_{\text{hard}}$.

In what follows, by special cases, unless otherwise indicated, we mean simple, semisimple, or numerically simple special cases. We now define the pairwise dissimilarity function.

Definition 2.6. *A pairwise dissimilarity function* is a function that measures the difference between instances according to some criterion.

Given a simple special case $P^{\alpha}_{\text{simple}}$ and the corresponding polynomial or pseudopolynomial algorithm, for any $B \in P^{\alpha}_{\text{simple}}$ and $A \notin P^{\alpha}_{\text{simple}}$, the absolute (or relative) error of the objective function for A when the polynomial (pseudopolynomial) exact solution for B is applied to A will be used as the value of the dissimilarity function, denoted by τ, on the pair (A, B). The steps of the pairwise similarity method are described below.

Note that different values, such as relative error, rate of convergence of the method, convergence scores of several methods, number of operations, etc., can be used to construct dissimilarity functions in the pairwise similarity method. Let us move on to the main definitions assuming that the problem under consideration is a linear programming problem with one objective function.

In the pairwise similarity method, a solution X_B to the instance B obtained using a polynomial (or pseudopolynomial) algorithm is applied to an instance A, for which it may violate the internal constraints. Depending on how the solution X_B is applied, we consider three strategies.

1. Evaluation of instance A: using the obtained solution X_B to evaluate the solution X_A, regardless of the feasibility of the resulting solution.
2. Relaxation of instance A: using the obtained solution X_B to evaluate the solution X_A, taking into account which restrictions of instance A are violated when X_B is applied to A. To implement this strategy, it is necessary to check the resulting solution for feasibility. That is, if instance B is a relaxation of instance A, then a procedure for checking the satisfaction of constraints is needed in order to determine which particular restrictions of instance A are violated by solution X_B.
3. Constructing a valid solution for instance A using the obtained solution X_B as an approximate solution to instance A. To do this, it is necessary to check the feasibility of X_B and, if some constraints are violated, find the new solution X^*_B closest to X_B among the solutions that satisfy all the constraints of A. Thus, to obtain $X_A = X^*_B$, it is necessary to construct a function transforming X_B into X^*_B.

Thus, the pairwise similarity method consists of the following steps.

1. Find one or more special cases of the problem. For many discrete optimization problems, some special cases are known. Below we present guidelines for finding new special cases.
2. Construct a pairwise dissimilarity function that allows one to estimate the distance from any instance to each of the special cases. An algorithm for building such a function is proposed below.
3. Given an instance A, find the closest special case using the chosen pairwise dissimilarity function.
4. Find the solution to the closest to A instance that belongs to a special case.
5. Estimate the guaranteed error of the obtained solution using the chosen pairwise dissimilarity function.
6. Choose a strategy for applying the solution of Item 4 to instance A.
   - For the "Evaluation of instance A" strategy, we use X_B without adjustment.

– For the strategy "Relaxation of instance A", we use X_B, taking into account information about whether the constraints of the original instance are violated, and if so, which ones.
– For the strategy "Constructing a valid solution for instance A", we check the obtained solution for feasibility, and if some restrictions are violated, then use the function transforming X_B into $X_A = X_B^*$.

Finding special cases is a non-trivial task, the solution of which is based on studying the special properties of the problem under consideration. For example, for the classes of problems where the graph included in a constraint is a tree, there are often fast exact algorithms based on classical tree traversal algorithms. Expanding the set of special cases allows one to more accurately determine the absolute error of the original instance. We now give some guidelines that help find special cases.

The most obvious way is to expand the known special cases by slightly changing the original pattern. Section 3.1 gives an instance of such an extension by adding additional edges to the trees to form cycles.

The second way is based on the following statement. If for problem P there exists an approximate algorithm $\tilde{\mathbf{A}}$, then among the instances of P one can single out a subset of instances P^* for which this algorithm provides exact solutions. To implement this guideline, it is necessary to solve a number of randomly selected instances of problem P with different parameters using the exact algorithm $\mathbf{A}$ and the approximate algorithm $\tilde{\mathbf{A}}$. Next, a subset of instances $P^* \subset P$ for which exact solutions (or solutions with a guaranteed error) were obtained by the approximate algorithm $\tilde{\mathbf{A}}$ must be selected. Note that the subset of instances P^* is, by definition, a numerically simple special case. Then it is necessary to determine whether there is a subset of instances $P^{**} \subseteq P^*$ for which some pattern α can be formulated. If the pattern α is found, then the resulting set of instances P^{**} is, by definition, a simple or semisimple special case.

The third way involves an exact combinatorial algorithm $\mathbf{A}$, for which the rate of convergence is investigated. If for problem P it is possible to select a subset of instances $P^* \subset P$ for which algorithm $\mathbf{A}$ converges fast enough, then one may try to construct a heuristic algorithm $\tilde{\mathbf{A}}$ based on $\mathbf{A}$, which provides an exact solution or an approximate solution with a guaranteed error for all instances in P^*. Note that the concept of fast convergence of an algorithm in this context is used informally, since the formalization depends on a specific problem. To implement this guideline, it is necessary to randomly select a number of instances of P with different parameters and solve them using $\mathbf{A}$. Next, we should single out a subset of instances $P^* \subset P$ for which $\mathbf{A}$ obtains exact solutions fairly quickly. Then it is necessary to build a heuristic algorithm based on the steps of $\mathbf{A}$, which are carried out in the course of solving the instances in P^*. As a result, we get P^* as a numerically simple special case. Further, similarly to the second way, we look for a pattern α that describes some subset of instances $P^{**} \subseteq P^*$. If such a pattern has been found, then P^{**} is a simple special case.

We now turn to the construction of a pairwise dissimilarity function. This includes the following steps.

1. Find a special case P^*.
2. Fix all parameters of the P^* problem, except for one parameter q.
3. Solve some set of instances of the problem P^* for various q.
4. Determine the relationship β^q between q and the exact value of the objective function.
5. Construct a pairwise dissimilarity function ρ^q that approximately represents β^q.
6. Repeat steps (2)–(5) for the other problem parameters.
7. As the resulting pairwise dissimilarity function, consider the sum $\rho = \sum_{q=1}^{n} \rho^q$ of partial functions for all parameters of problem P.

An example of the application of this procedure is the variation of the cyclomatic number of a graph in Section 3.1, where the pairwise similarity method is used to build an analytical estimate with a limited absolute error.

Note that, due to the complexity of the problem, the pairwise dissimilarity function cannot always be found analytically. In this case, it is proposed to evaluate the pairwise dissimilarity function numerically. For instance, in Section 3.2 the pairwise dissimilarity function is evaluated using computational experiments. It is proposed to majorize from above and tabulate the numerical values of the pairwise dissimilarity function.

3. APPLICATIONS OF THE PAIRWISE SIMILARITY METHOD

3.1. Majority domination problem

In this section, the pairwise similarity method is illustrated by the its application to the problem of finding the strict majority domination number for a graph with cycles within the two-tier voting model. The pairwise dissimilarity function is built using the procedure described in Section 2. The special case is constructed by extending the well-known special case, where the constraint graph is a tree.

In social and technical systems, two-tier majority procedures are often used to make decisions. In such procedures, at the first stage, majority voting is carried out in local groups of agents, at the second stage, the results of group voting are aggregated, also using majority, into the final decision [3]. One of the central questions in the analysis of two-tier procedures is: "For what minimum proportion of agents that support a proposal can it ultimately be accepted by this procedure?" For a given graph of information connections between the agents, the minimum difference between the number of agents that support an accepted proposal and the number of agents that do not support it is called the strict majority domination number of the graph.

We will use the following notation. $G = (V, E)$, $|V| = n$, is a graph, each vertex of which has a loop. Vertex $v \in V$ has the *set of neighbors* $N_v = \{w \in V : \{v, w\} \in E\}$, which includes v. We will write $N_v(G)$ when several graphs share the same vertex set. An *opinion function* $f : V \to \{-1, 1\}$ associates an opinion with each vertex. The opinion function is extended to subsets $W \subseteq V$ as the sum of the opinions of the vertices that belong to W: $f(W) = \sum_{v \in W} f(v)$. Each vertex $v \in V$ *votes* "*for*" if $f(N_v) > 0$ or "*against*" otherwise. A pair $(G, f) \equiv G^f$ will be called a *configuration*. Let $V^+ = \{v \in V : f(N_v) > 0\}$ be the set of vertices voting "for".

Definition 3.1. Given a configuration, a proposal put to a vote is *accepted* if $|V^+| > |V|/2$ and *rejected* otherwise; acceptance or rejection are the possible *results of the vote*. An opinion function for which a proposal is accepted is called a *strict majority function for* G.

Definition 3.2. The *strict majority domination number* of a graph G is $\gamma(G) = \min_{f \in \mathcal{F}^+} f(V)$, where $\mathcal{F}^+$ is the set of strict majority functions for G. A strict majority function f will be called *optimal* whenever $f(V) = \gamma(G)$.

Finding the strict majority domination number of a given graph G and the corresponding optimal opinion function is a discrete optimization problem whose solution space is given by 2^n possible opinion functions. For the case of an arbitrary graph G, this is an NP-complete problem [1], however, there are special cases for which a polynomial algorithm that solves it exists. In particular, if G is a tree, then an optimal opinion function can be constructed in polynomial time using a dynamic programming algorithm [9].

The problems of finding the strict majority domination number on trees act as *a simple special case* in the application of the pairwise similarity method described below. In accordance with the general scheme of the method given in Section 2, we will show that the case of graphs with an upper bounded cyclomatic number is a semisimple special case for

the problem of finding the strict majority domination number of a graph, that is, it allows the construction of an approximate solution with a guaranteed absolute error by reduction to the case of a tree. Thus, a limited number of independent cycles is the pattern α around which the application of the pairwise similarity method can be developed.

In Theorem 3.1 below, we prove that for any spanning tree T of graph G having k cycles, $\gamma(G) \geq \gamma(T) - 4k$ holds. It also follows from the proof that knowing the optimal opinion function f of a spanning tree T, one can construct a strict majority function $\tilde{f}$ for the graph G such that $\tilde{f}(V) \leq \gamma(T) + 2k$, which follows that the error of the approximate solution $\tilde{f}(V) - \gamma(G)$ does not exceed $6k$. This theorem allows us to reduce the search for an approximate solution for G to the search for an optimal opinion function for a tree T, which can be done in polynomial time [1], while the guaranteed error of the solution depends only on the number of cycles in the graph, not on its order or size.

We first introduce a technical notion and prove an auxiliary result.

Definition 3.3. Let configurations G^f and $G'^{\tilde{f}}$ share vertex set V. We say that the *voting conditions of* $v \in V$ *in* $G'^{\tilde{f}}$ *are no worse than in* G^f if v has at least as many neighbors in $G'^{\tilde{f}}$ with opinion $+1$ than in G^f and no more neighbors in $G'^{\tilde{f}}$ with opinion -1 than in G^f.

Lemma 3.1:
Let G be a graph with $E(G) \neq \emptyset$ and H be the graph obtained from G by removing some edge uv. Then

$$-4 \leq \gamma(G) - \gamma(H) \leq 2. \tag{3.1}$$

Proof

Suppose f is an optimal opinion function for H; therefore, $f(V) = \gamma(H)$. Let us bound $\gamma(G)$ from above and below. To do this, consider the cases where f is not a strict majority function for G along with modifications of f that make the resulting function $\tilde{f}$ a strict majority function for G. Then $\tilde{f}(V)$ is an upper bound for $\gamma(G)$.

First consider the cases where vertices u and/or v vote "for" in H^f, while without this the proposal would be rejected. In such cases, after adding edge uv, the voting conditions for u and/or v may worsen, which may lead to the violation of the strict majority condition for f in G^f. If these vertices vote "against" in H^f, then changing the voting conditions for them with the addition of edge uv does not cancel the acceptance of the proposal and preserves the strict majority condition for f in G^f.

1. Let $f(u) = f(v) = 1$. In this case, adding a neighbor with opinion $+1$ will not worsen the voting conditions for either vertex. Therefore, f remains a strict majority function on G, but may cease to be optimal. Hence $\gamma(G) \leq \tilde{f}(V) = f(V) = \gamma(H)$.
2. Let $f(u) = f(v) = -1$. Then the addition of edge uv worsens the voting conditions for both u and v.
   - If u and v vote "for" in H^f and $f(N_u(H)) = 1$ or $f(N_v(H)) = 1$, then after adding edge uv, the vote of one of the vertices changes to "against". As a result, f may cease to be a strict majority function. To obtain a function $\tilde{f}$ having this property, it is sufficient to change the opinion of u or v (let it be u) from -1 to $+1$. Then $\tilde{f}(N_u(G)) = f(N_u(H))$, while $\tilde{f}(N_v(G)) = f(N_v(H)) + 1$. Thus, after adding edge uv and setting $\tilde{f}(u) = f(u) + 2$, the votes are preserved and so $\tilde{f}$ is a strict majority function for G. Hence $\gamma(G) \leq \tilde{f}(V) = f(V) + 2 = \gamma(H) + 2$.
   - If u or v votes "for" and the other vertex votes "against" in H^f, then to define a strict majority function $\tilde{f}$ for G, it suffices to set $\tilde{f}(u) = +1$ or $\tilde{f}(v) = +1$. In this case also $\gamma(G) \leq \tilde{f}(V) = f(V) + 2 = \gamma(H) + 2$.

3. Let $f(u) = -1$, $f(v) = 1$. For u, adding edge uv does not worsen the voting conditions. If v votes "against" in H^f, then f remains a strict majority function after adding edge uv, since the number of votes "for" does not decrease. If v votes "for", then obtaining a strict majority function $\tilde{f}$ for G reduces to setting $\tilde{f}(u) = +1$, which implies $\tilde{f}(V) = f(V) + 2$. Therefore, $\gamma(G) \leq \tilde{f}(V) = \gamma(H) + 2$.

Now let f be an optimal strict majority function for G, which implies $f(V) = \gamma(G)$. Let us bound $\gamma(H)$ from above by transforming f into a strict majority function $\tilde{f}$ for H.

1. Let $f(u) = f(v) = -1$. Removing edge uv does not worsen the voting conditions for u or v. Hence f remains a strict majority function for H and $\gamma(H) \leq f(V) = \gamma(G)$.
2. Let $f(u) = f(v) = 1$. Then removing edge uv worsens the voting conditions for both u and v.
   - If u or v votes "for" in G^f (let it be u), while the other vertex votes "against", then f may cease to be a strict majority function for H. This may happen if $f(N_u(G)) = 1$, in which case $f(N_u(H)) = 0$. To obtain a strict majority function $\tilde{f}$ for H, it suffices to change the opinion of one of the neighbors of u from -1 to 1. This can always be done, since $f(N_u(H)) = 0$ and $f(u) = +1$ imply the existence of a neighbor of u with opinion -1. Therefore, $\tilde{f}(V) = f(V) + 2$.
   - If both u and v vote "for" in G^f, then f may cease to be a strict majority function for H if $f(N_u(G)) = 1$ or $f(N_v(G)) = 1$ (or both). Then a strict majority function $\tilde{f}$ for H can be obtained by changing the opinion of one neighbor of u or/and one neighbor of v from -1 to $+1$, as in the previous case. Then $\tilde{f}(V) \leq f(V) + 4$. Thus, $\gamma(H) \leq \tilde{f}(V) \leq \gamma(G) + 4$.
3. Let $f(u) = 1$, $f(v) = -1$. Removing edge uv improves the voting conditions for u and worsens them for v. If v votes "for" in G^f, then this may lead to its vote "against" in H^f. As a result, f may cease to be a strict majority function for H. To obtain a strict majority function $\tilde{f}$ for H, it suffices to set $\tilde{f}(v) = +1$, which yields $\tilde{f}(N_v(H)) = f(N_v(G)) + 1$ and $\tilde{f}(N_u(H)) = f(N_u(G)) + 1$. Then $\gamma(H) \leq \tilde{f}(V) = f(V) + 2 = \gamma(G) + 2$, since $f(V) = \gamma(G)$.

Collecting the obtained inequalities, we get (3.1). □

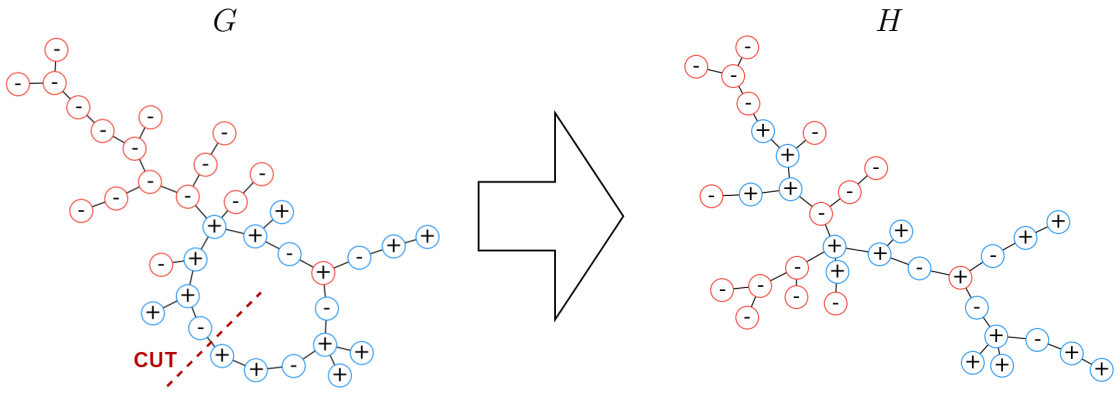

Fig. 3.1. Graph G and graph H obtained from G by removing one edge such that $\gamma(G) - \gamma(H) = -4$, since $\gamma(G) = -7$ and $\gamma(H) = -3$.

Inequality (3.1) is sharp. Examples of graph G and graph H obtained from G by removing one that satisfy that $\gamma(G) - \gamma(h) = -4$ and $\gamma(G) - \gamma(h) = 2$ are shown in Fig. 3.1 and Fig. 3.1, respectively.

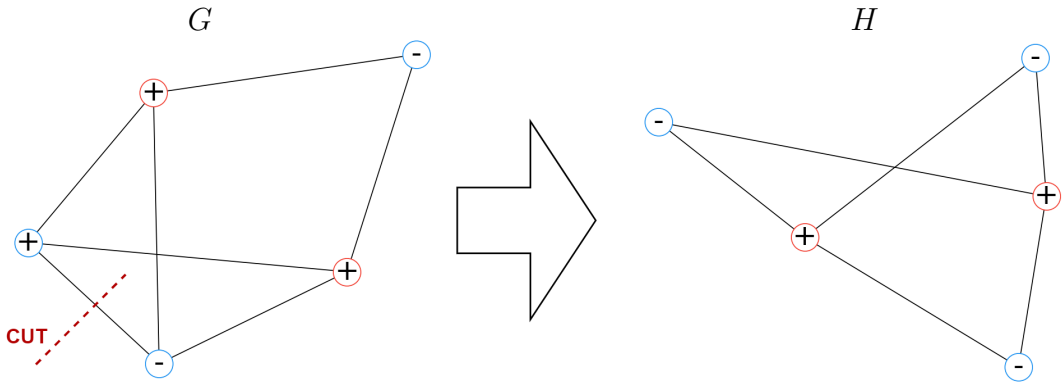

Fig. 3.2. Graph G and graph H obtained from G by removing one edge such that $\gamma(G) - \gamma(H) = 2$, since $\gamma(G) = 1$ and $\gamma(H) = -1$.

Theorem 3.1:
Let a connected graph G have cyclomatic number k, and let T be its arbitrary spanning tree. Then $\gamma(T) - 4k \leq \gamma(G) \leq \gamma(T) + 2k$.

Proof
By the definition of cyclomatic number, tree T is obtained from G by removing k edges. Deleting these edges one by one generates a sequence of graphs $G := H_0, H_1, ..., H_k =: T$. The proof of the theorem reduces to applying Lemma 3.1 to pairs of graphs H_{i-1}, H_i, $i = 1, ..., k$ and summing the resulting inequalities (3.1) for them. □

Theorem 3.1 defines a pairwise dissimilarity function, according to which each spanning tree of a graph G with cyclomatic number k is at distance k from G. A spanning tree of G can be found via breadth-first search or depth-first search algorithms, which have linear time complexity. On the other hand, spanning trees refer to a simple special case of the strict majority domination number problem, which allows one to find the strict majority domination number for any of them in polynomial time $O(n^2)$ and use the tree's optimal strict majority function to construct a strict majority function for the graph G, which gives an approximate solution to the original problem.

A strict majority function for a graph G with cyclomatic number k can be found as follows:

1. Find a spanning tree T of G. Determine an optimal strict majority function for T using the polynomial algorithm [9].
2. Sequentially add k edges to T until G is obtained:
   - Add an edge that belongs to $E(G) \setminus E(T)$. With the current opinion function, check if $|V^+| > |V|/2$. If this is true, proceed by adding the next edge.
   - Otherwise, change the opinion -1 of a vertex incident to the new edge to $+1$.

In accordance with the general scheme of the pairwise similarity method, as an approximate strict majority domination number of G, the best solution constructed on the basis of the optimal strict majority functions of the spanning trees of G should be taken. In practice, one can limit oneself to constructing several randomly selected spanning trees.

3.2. Maximum cut problem

In this section, the pairwise dissimilarity function based on the relative error is estimated numerically. For this, an estimate of the error is constructed, based on the reduction of problem instances to a simple special case. We use bipartite graphs as a simple special case for the maximum cut problem. First, we introduce the main definitions related to this problem.

Definition 3.4. A *bipartite graph* is a graph whose set of vertices can be divided into two parts in such a way that each edge of the graph connects a vertex of one part with a vertex of the other part.

Definition 3.5. A *cut* is a partition of the vertices of a graph into two subsets. Any cut determines a *cut-set*, the set of edges that have one endpoint in each subset of the partition. These edges are said to *cross* the cut.

Definition 3.6. In the case of an unweighted graph, the *size* or *weight* of a cut is the number of edges crossing the cut; in the case of a weighted graph, the *weight* of a cut is the sum of the weights of the crossing edges.

Definition 3.7. A *maximum cut* of a graph is a cut whose size is at least the size of any other cut (an example is shown in Fig. 3.3).

The maximum cut problem for unweighted undirected graphs can be formulated as follows. Given a connected graph G, find a subset S of its vertices such that the number of edges connecting S with $V(G) \setminus S$ is maximum.

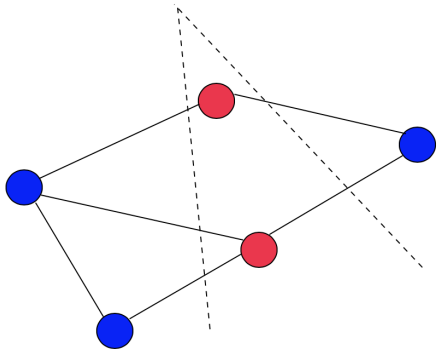

Fig. 3.3. Maximum cut of a graph

This problem is NP-hard [5]. Consider the maximum cut problems on bipartite graphs (Definition 3.4) as a special case. For them, the solution is trivial: the maximum cut includes all edges. It can be easily checked in linear time in the number of edges whether a graph is bipartite, therefore, by Definition 2.2, bipartite graphs determine a simple special case for the maximum cut problem.

The following simple depth-first search algorithm finds an approximate solution to the maximum cut problem; this solution is exact for bipartite graphs. The error of this algorithm gives rise to a dissimilarity function, which is an upper-bound estimate of the error.

The input of Algorithm 1 is an arbitrary undirected connected graph $G = (V, E)$. By removing edges, the algorithm selects a bipartite graph $G_b = (V, E_b)$. It works recursively and paints the vertices in two colors, red and blue for definiteness. First, we color an arbitrary vertex blue and the adjacent vertices a different color (in this case, red).

Further, the algorithm recursively repeats this procedure on all adjacent vertices that were painted at the current iteration. If the current and an adjacent vertices have the same color, then instead of repainting the latter one, we remove the edge between them. As a result, the algorithm returns a bipartite graph with blue and red vertices, the colors of adjacent vertices of which differ. The complexity of the algorithm is $O(|V|^2)$, since the algorithm passes through all the vertices of the graph, and for each vertex, its adjacent vertices are checked.

According to the general concept of dissimilarity function (Section 2), the dissimilarity based on the relative error between the maximum cut problems A_G with graph G and A_H

Algorithm 1 MAX-CUT heuristic algorithm

function MAX-CUT_HEURISTICS(v)
  A – adjacency matrix of the connected graph
  $color$ = [0 for v in V] # 0 – not colored, -1 – first color, 1 – second color
  v := Random(V) # choose an arbitrary vertex
  $color[v]$ = 1 # paint first vertex
  COLOR_ADJACENT_VERTICES(v)

  return $\sum_{u,v} \frac{A[u,v]}{2}$
end function
procedure COLOR_ADJACENT_VERTICES(v)
  for u in V **do**
    if $A[u,v] = 1$ and $color[u] = color[v]$ **then**
      $A[u,v]$:= 0 #remove the edge
    else if $A[u,v] = 1$ and $color[u] = 0$ **then**
      $color[u]$:= $-color[v]$ #paint the vertex in the opposite color
      COLOR_ADJACENT_VERTICES(u)
    end if
  end for
end procedure

with a bipartite graph H such that $E(H) \subseteq E(G)$ can be defined as

$$\tau(A_G, A_H) = \frac{f(A_G) - f(A_H)}{f(A_G)},$$

where $f(A_G)$ is the optimal value of the objective function for A_G. In what follows, for simplicity, we will use $f(G)$ as an abbreviation for $f(A_G)$.

For the above algorithm, it is not easy to obtain the upper bound for the dissimilarity between A_G and A_H, where H is the bipartite graph provided by the algorithm, analytically. However, this bound can be approximated by numerical experiments.

In such experiments, graphs $G_i(N,p)$ are randomly generated with two parameters: the number of vertices N and the density p (the ratio of the number of edges in the graph to the maximum possible number of edges $N(N-1)/2$). That is, the probability that there is an edge between any two vertices is p. All edge weights are equal to 1. Each point in the figures below is the average over 100 random instances.

To estimate the upper bound of the pairwise dissimilarity function, the average relative error and the maximum relative error are used:

$$\delta_{\text{av}}(N,p) = \frac{1}{K}\sum_{i=1}^{K} \frac{f(G_i(N,p)) - \hat{f}(G_i(N,p))}{f(G_i(N,p))};$$

$$\delta_{\max}(N,p) = \max_{i\in\{1,\dots,K\}} \frac{f(G_i(N,p)) - \hat{f}(G_i(N,p))}{f(G_i(N,p))},$$

where K is the number of instances with given parameters, $\hat{f}(G_i(N,p))$ is the value of the objective function of the algorithm, and $f(G_i(N,p))$ is the optimal value of the objective function.

Fig. 3.4 shows the dependence of the average relative error $\delta_{\text{av}}(N,p)$ of the objective function on the number of vertices in a graph with a density $p = 0.6$. It can be seen that with an increase in the number of vertices, the relative error decreases.

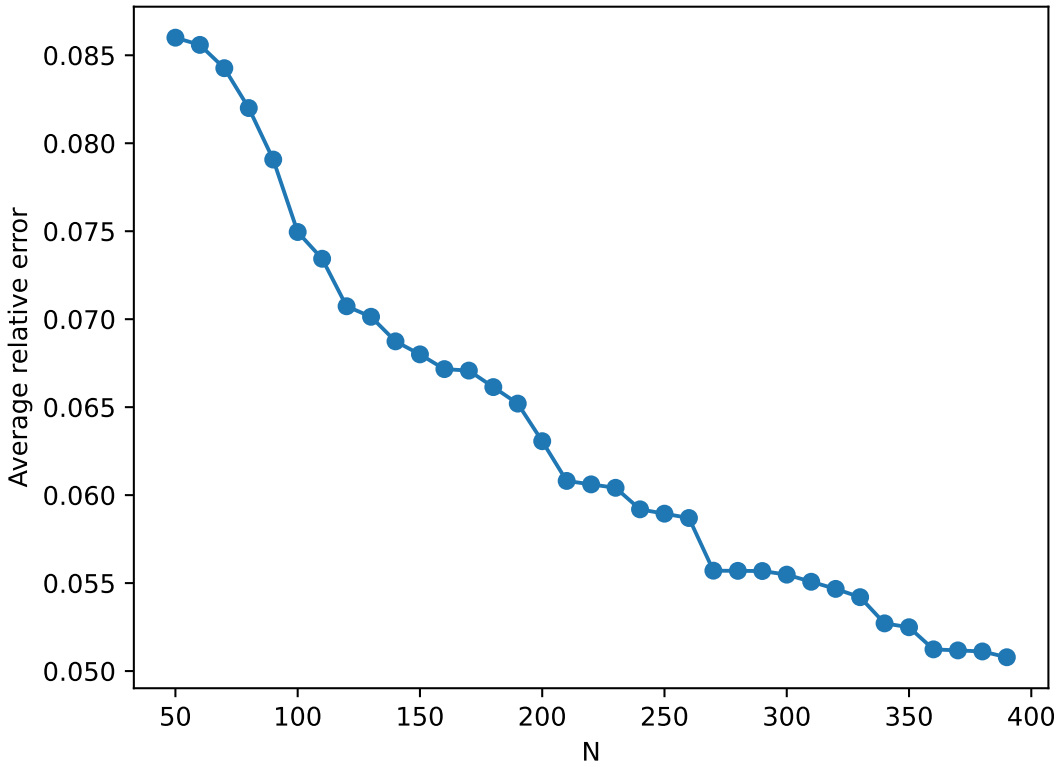

Fig. 3.4. Dependence of the relative error of the objective function on the number of vertices.

Fig. 3.5 shows the dependence of the average relative error $\delta_{\text{av}}(N, p)$ of the objective function on the density of a graph with a constant number of vertices $N = 150$. It can be seen that with increasing density the relative error decreases.

Fig. 3.6 shows the dependence of the average relative error $\delta_{\text{av}}(N, p)$ of the objective function on the density of a graph and on the number of vertices. It can be seen that the density of the graph has a much stronger effect on the relative error than the number of vertices.

Fig. 3.7 shows the dependence of the maximum relative error $\delta_{\max}(N, p)$ of the objective function on the density of a graph. The shape of this curve is very similar to that of the average relative error one.

Fig. 3.8 shows the dependence of the maximum relative error $\delta_{\max}(N, p)$ of the objective function on the number of vertices. This curve is also similar to the curve of the average relative error, but it is less robust.

Fig. 3.9 shows the dependence of the maximum relative error $\delta_{\max}(N, p)$ of the objective function on the density and number of vertices. It is not monotonic in the region of small numbers of vertices, since for such numbers, the solution is very unstable.

Since we choose a random vertex to start the algorithm, the algorithm works in a non-deterministic way. Obviously, it is necessary to remove as few edges as possible from the initial graph, since the number of remaining edges determines the value of the objective function. Therefore, in the general case, appropriate heuristics can be used to select the first vertex and study the influence of this choice on the size of the resulting bipartite graph.

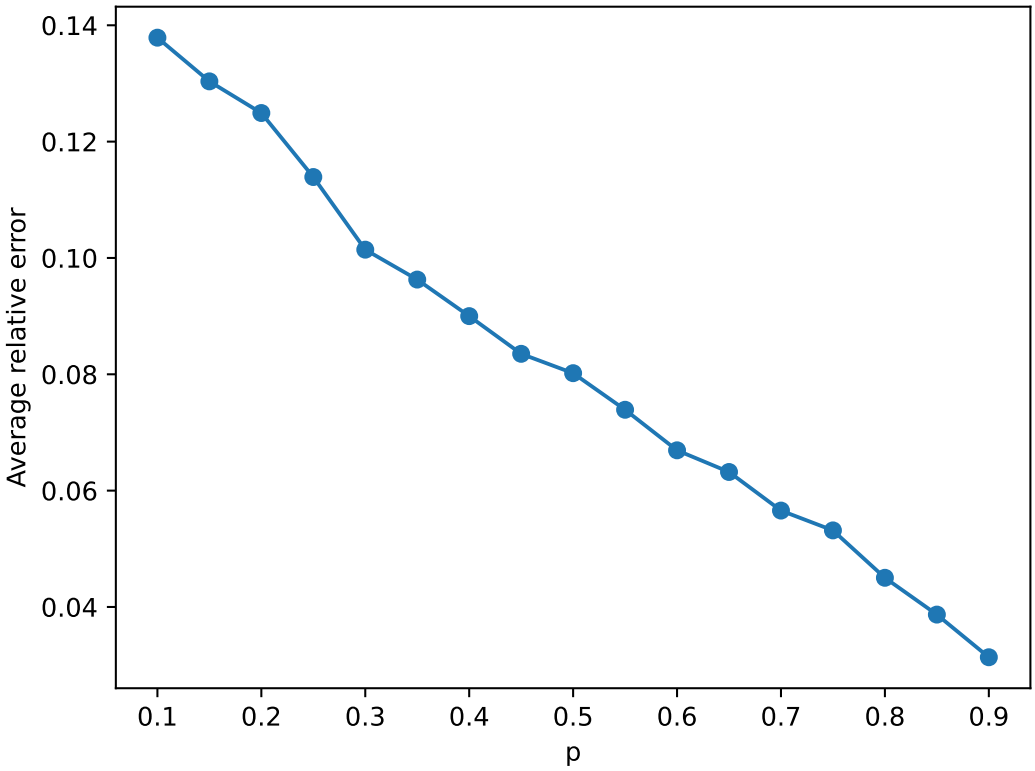

Fig. 3.5. The dependence of the relative error of the objective function on the density.

According to the general scheme of pairwise similarity method, it is suggested to run the heuristic Algorithm 1 several times and choose the graph with the least number of removed edges. In this case $\delta_{\text{av}}(N, p)$ can be used as an estimate for the upper bound for the dissimilarity, while $\delta_{\max}(N, p)$ is more appropriate if the algorithm was run once.

4. CONCLUSION

In this paper, a new pairwise similarity method is proposed for measuring the quantitative complexity of NP-hard problems. This method, which generalizes the metric approach, allows one to compare problem instances of different dimensions and with different constraint structures. This, in turn, makes it possible to use algorithms developed for some special cases of problems as heuristic algorithms for a wider class of problems. Using the pairwise dissimilarity function, one can estimate the error of the obtained solutions.

Since the pairwise similarity method includes two non-trivial tasks, namely, searching for new special cases and defining the pairwise dissimilarity function, the paper provides recommendations for solving these problems.

The proposed method is illustrated for two optimization problems. For the problem of finding the strict majority domination number of a graph with cycles in the framework of a two-level voting model, the pairwise similarity method is applied in the case where it is possible to analytically estimate the absolute error as specified in Section 2. For the maximum

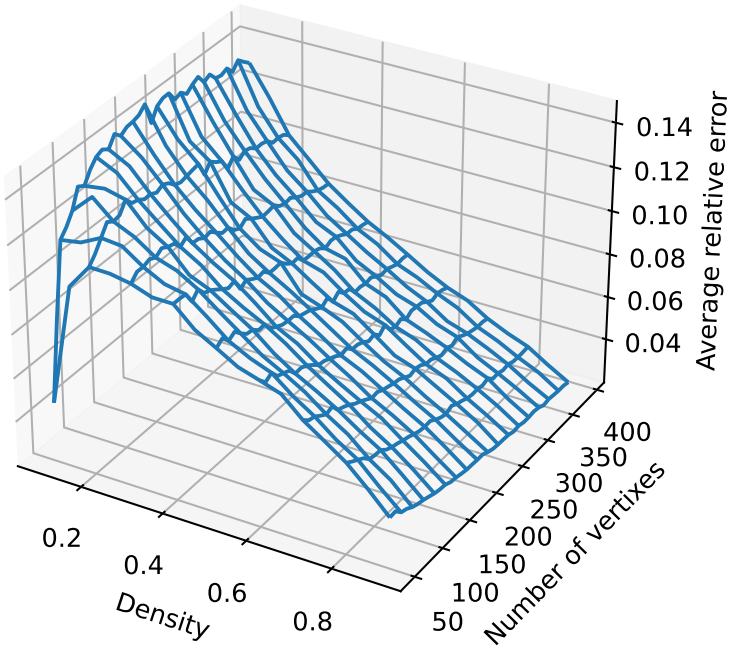

Fig. 3.6. The dependence of the relative error of the objective function on the density and number of vertices.

cut problem, the pairwise similarity method is applied in the case where it is not possible to estimate the error analytically and the estimates are based on numerical experiments.

In the future, it is planned to apply the pairwise similarity method to other well-known optimization problems on graphs and to study a number of variations of similarity criteria. It is also planned to develop a system of complexity maps for the most important similarity criteria.

ACKNOWLEDGEMENTS

The research was supported by RSF (project No. 22-71-10131). Pavel Chebotarev is supported by the European Union (ERC, GENERALIZATION, 101039692). The results of Sections 1, 2, Subsection 3.2 and Theorem 3.1 were obtained within the RSF grant. Views and opinions expressed are those of the authors only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

REFERENCES

[1] I. Broere, J. H. Hattingh, M. A. Henning, and A. A. McRae. Majority domination in graphs. *Discrete Mathematics*, 138(1–3):125–135, 1995.

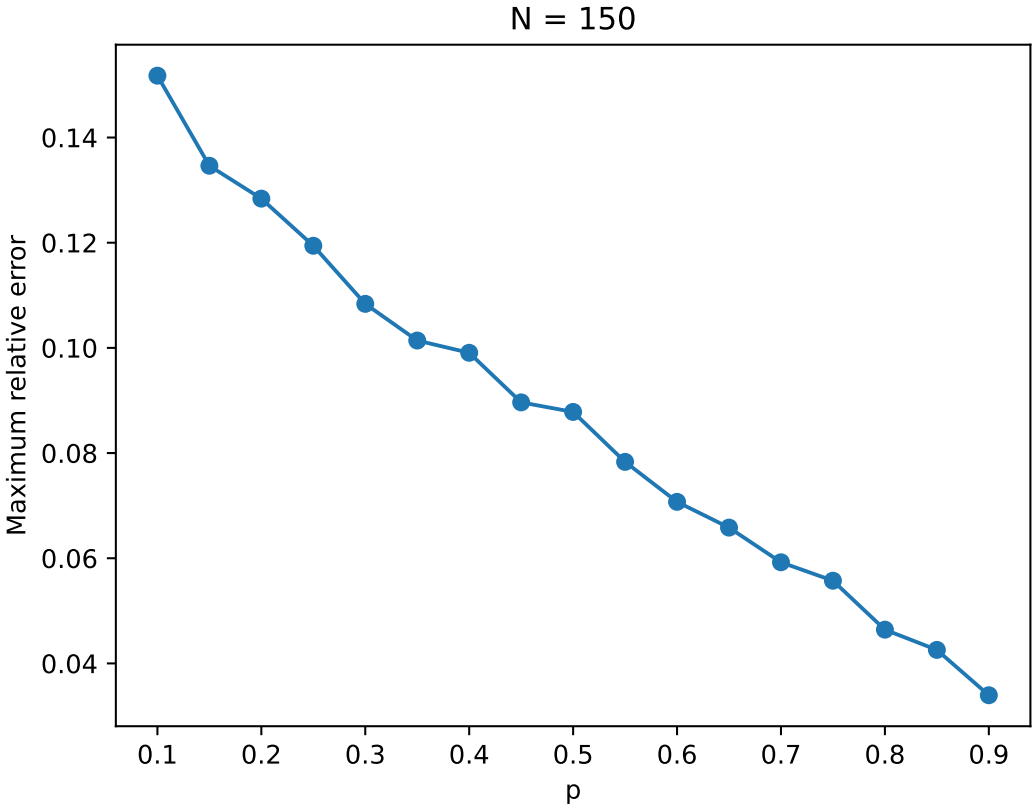

Fig. 3.7. The dependence of the maximum relative error of the objective function on the density.

[2] E. Bukueva, I. Kudinov, and D. Lemtyuzhnikova. Analysis of the feasibility to use metric approach for np-hard makespan minimization problem. *IFAC-PapersOnLine*, 55(10):2898–2901, 2022.

[3] P. Chebotarev and D. Peleg. The power of small coalitions under two-tier majority on regular graphs. Preprint [math.OC] 2212.03394, arXiv, 2022. https://doi.org/10.48550/arXiv.2212.03394.

[4] T. E. Cheng, A. Lazarev, and D. Lemtyuzhnikova. A metric approach for the two-station single-track railway scheduling problem. *IFAC-PapersOnLine*, 55(10):2875–2880, 2022.

[5] R. M. Karp. *Reducibility among combinatorial problems*. Springer, 2010.

[6] A. Lazarev, D. Lemtyuzhnikova, N. Pravdivets, and F. Werner. Polynomially solvable subcases for the approximate solution of multi-machine scheduling problems. In *Advances in Optimization and Applications: 11th International Conference, OPTIMA 2020, Moscow, Russia, September 28–October 2, 2020, Revised Selected Papers 11*, pages 211–223. Springer, 2020.

[7] A. A. Lazarev, D. V. Lemtyuzhnikova, and N. A. Pravdivets. Metric approach for finding approximate solutions of scheduling problems. *Computational Mathematics and Mathematical Physics*, 61:1169–1180, 2021.

[8] A. A. Lazarev, D. V. Lemtyuzhnikova, and F. Werner. A metric approach for scheduling problems with minimizing the maximum penalty. *Applied Mathematical Modelling*, 89:1163–1176, 2021.

[9] H.-G. Yeh and G. J. Chang. Algorithmic aspects of majority domination. *Taiwanese Journal of Mathematics*, pages 343–350, 1997.

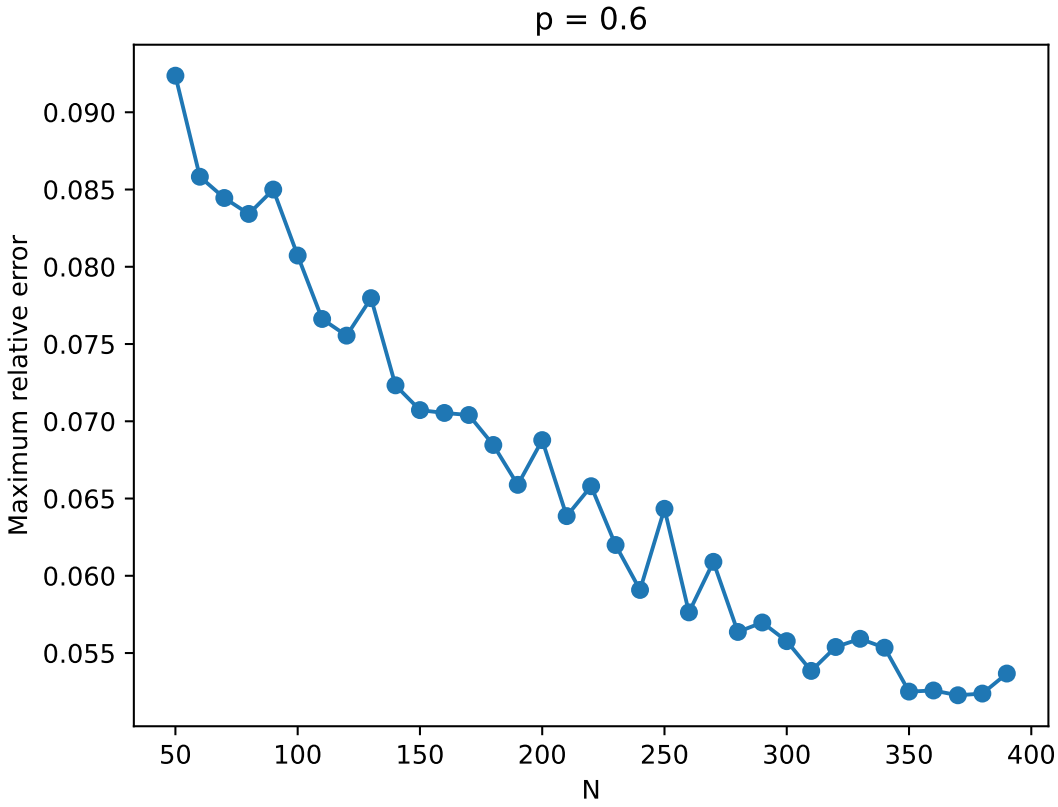

Fig. 3.8. The dependence of the maximum relative error of the objective function on the number of vertices.

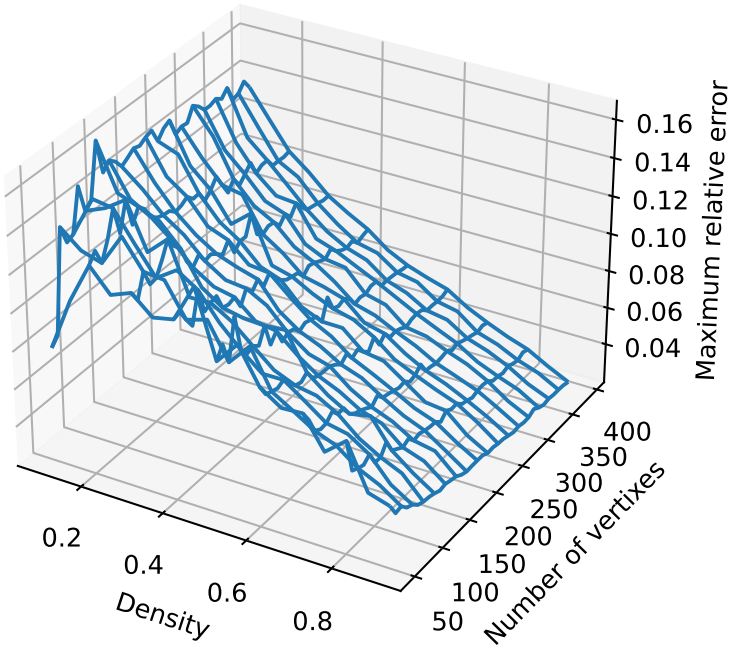

Fig. 3.9. The dependence of the maximum relative error of the objective function on the density and number of vertices.

