Hybrid Techniques of Combinatorial Optimization with Application to Retail Credit Risk Assessment

Stjepan Oreski*

Bank of Karlovac, I. G. Kovačica 1, 47000 Karlovac, Croatia
*Corresponding author: stjepan.oreski@kaba.hr

Abstract:

Hybrid techniques of combinatorial optimization are a growing research area, designed to solve complex optimization problems. In the first part of this paper, we focus on the methodological background of hybrid techniques of combinatorial optimization, paying special attention to the important concepts in the field of combinatorial optimization and computational complexity theory, as well as to hybridization strategies that are important in the development of hybrid techniques of combinatorial optimization. According to the presented relations among the techniques of combinatorial optimization, the strategies of combining them and the concepts for solving combinatorial optimization problems, this paper presents an example of the hybrid technique for feature selection and classification in credit risk assessment. This study emphasizes the importance of hybridization as a concept of cooperation among metaheuristics and other optimization techniques. The importance of such cooperation is confirmed by the results that are presented in the experimental part of the paper, which were obtained on a German credit dataset using the hybrid technique of combinatorial optimization based on a low-level relay strategy. The experimental results show that the proposed method outperforms, on the same dataset, the methods presented in the literature in terms of the average prediction accuracy.

Keywords:

Hybrid Technique; Combinatorial Optimization; NP-hard Problem, Heuristic; Diversification; Intensification

1. INTRODUCTION

Combinatorial optimization (CO) solves optimization problems that have a pronounced combinatorial or discrete structure [1]. Hybrid techniques of combinatorial optimization combine different techniques to solve combinatorial optimization problems efficiently. CO has an increasing importance because of the large number of practical problems that can be formulated and solved as CO problems. The most interesting discrete optimization problems are the NP-hard problems, which are the central subject of this paper. Thus, unless P = NP, there are no efficient algorithms known at the present time that can find optimal solutions to such problems, where we follow the convention that an efficient algorithm is one that runs in time bounded by a polynomial on its input size [2]. Heuristic (mostly metaheuristic) algorithms...
are usually considered and are applied to solve the previously mentioned problems. These algorithms very
often provide very good or (almost) optimal solutions in a relatively short time, which, however, comes
with the price of no guarantee of quality [3].

The main feature of the exact algorithms in solving these problems is that they give optimal solutions,
but in the worst case, they accomplish this goal in an exponential execution time. Despite many studies
and the effort of scientists and practitioners, exact algorithms are generally applicable only to cases of
limited size, even though that size has definitely become larger over time. In contrast, the performances of
heuristics are usually better as a problem gets bigger, which makes them especially attractive for practical
use. From that point of view, exact and heuristic algorithms are completely opposite, and no one can have
optimality and wide applicability at the same time.

Heuristic algorithms are developed as an answer to the impossibility of exact algorithms in solving
many problems. In the case of NP-hard problems, we sacrifice optimality in favor of solutions that are
good enough and that are calculated in an efficient way. Trading with optimality in favor of adaptable
solving is the paradigm of heuristic algorithms [2].

Heuristic algorithms (heuristic techniques) for solving CO problems cover a range from simple heuristic
algorithms for solving the problem at hand to different metaheuristic techniques. Recently, heuristic
techniques, mainly metaheuristics, have been taken in combination with other techniques to form hybrids.
There are many studies that present an overview of hybrid techniques and some strategies for the
combination of different techniques when developing hybrid metaheuristic techniques. A few of the most
thorough and/or the more recent studies are [4–6]. However, there is no study that would give a brief but
comprehensive overview of the terms, concepts and strategies that are related to the hybrid techniques of
CO and that would present the concepts and demonstrate the strategies through an example. Therefore,
the aim of this study is to give a brief but comprehensive review of the terms, concepts and strategies that
are related to CO, and according to these, an empirical example that confirms the presented concepts and
strategies with the results.

The remaining sections of this paper are organized as follows. In Section 2, a comprehensive methodo-
logical background for the hybrid techniques of CO is presented. An empirical analysis of hybridization
strategies on a German credit dataset using a low-level relay strategy is presented in Section 3, followed
by the Conclusions.

2. METHODOLOGICAL BACKGROUND

This section is divided into four subsections. In Section 2.1, some relevant terms in the field of combi-
natorial optimization problems are defined. Section 2.2 addresses the computational complexity, while
Section 2.3 is dedicated to outlining the techniques for solving problems in combinatorial optimization
and the strategies for combining different techniques in developing hybrid metaheuristic techniques. The
fundamental concepts and principles that a technique for solving combinatorial optimization problems is
expected to fulfill are described in Section 2.4.

2.1 Defining Combinatorial Optimization Problems

Optimization problems are grouped basically into two categories: those whose solutions are coded with
real variables and those whose solutions are coded with discrete variables. Among the latter, we can find
a class of problems that are called combinatorial optimization problems. We are looking for an object
from a countable set. This object is usually an integer, a subset, a permutation or a graph structure [5].
Definition 2.1. A combinatorial optimization problem $P = (S, f)$ can be defined by
- a set of variables $X = x_1,...,x_n$;
- variable domains $D_1,...,D_n$;
- constraints among the variables $C$;
- an objective function $f$ to be minimized or maximized (depending on the problem),

where $f : D_1\times...\times D_n \rightarrow \mathbb{R}^+$; The set of all possible feasible assignments is $S = \{ s = \{(x_1,v_1),...,(x_n,v_n)\} \mid v_i \in D_i, s \text{ satisfies all of the constraints } C \}$

$S$ is usually called a search (or solution) space because each element of the set can be seen as a candidate solution. To solve a combinatorial optimization problem, one must find a solution $s^* \in S$ with a minimum objective function value, in other words, $f(s^*) \leq f(s) \forall s \in S$. Here, $s^*$ is called a globally optimal solution of $(S, f)$, and the set $S^* \subseteq S$ is called the set of globally optimal solutions [5, 7]. Because maximizing an objective function $f$ is the same as minimizing $-f$, in this work, we will address minimization problems without loss of generality.

The significance of optimization techniques in today’s society emerges from the common problems of processing a large amount of data, with the aim of making rational choices that, mathematically speaking, represent a globally optimal solution. The focus of the optimization methods is how to obtain a globally optimal solution and achieve the best results. Unfortunately, because most of those problems are NP-hard, there are no efficient algorithms for finding optimal solutions to such problems. It is, therefore, worth quoting [2] an old engineering slogan that says, Fast. Cheap. Reliable. Choose two. Similarly, if $P \neq$ NP, then we cannot simultaneously have algorithms that (1) find optimal solutions (2) in polynomial time (3) for any instance. At least one of these requirements must be loosened in any approach that addresses an NP-hard optimization problem.

An approach for solving problems depends, in many respects, on which request we are going to loosen. One approach loosens the for any instance requirement and finds polynomial-time algorithms for special cases of the problem at hand. This strategy is useful if the instances that one desires to solve fall into one of these special cases, but this circumstance is rarely the case.

A more common approach is to relax the requirement of polynomial-time solvability. The goal is then to find optimal solutions to problems by clever exploration of the full set of possible solutions to a problem. This strategy is often a successful approach if one is willing to take minutes, or even hours, to find the best possible solution; perhaps even more importantly, one is never certain that when the next input is encountered, the algorithm will terminate in any reasonable amount of time.

The most frequent approach is to loosen the find optimal solutions requirement. Loosening this criterion tends to involve as little compromise as is possible because we are attempting to obtain a solution that is close to optimal. Algorithms that give such solutions are approximate algorithms and can be divided [3] into (1) approximation algorithms and (2) other heuristics. The features of approximation algorithms are that they give a sub-optimal solution in polynomial time and guarantee that the solution is close to the optimum. The features of the heuristics are that they give an approximate solution while using fast and simple computing, which tends to be good enough, although there is no guarantee of accuracy. Approximation algorithms are the preferred types of approximate algorithms. Unfortunately, the use of approximation algorithms is relatively rare because they cannot be applied to many problems. Heuristics are the object of the studies of many researchers and practitioners because they solve the majority of the combinatorial optimization problems. A relatively new type of approximate algorithm has emerged, which is called metaheuristic algorithms. This name refers to heuristics that are not specifically developed for a certain problem.
Approximate algorithms are usually divided into constructive algorithms and algorithms for local search. Constructive algorithms generate a solution from scratch by adding to an initial empty set the components of the solution until a complete solution is accomplished. These algorithms are usually faster, but they often provide inferior solutions compared to algorithms for local search. Algorithms for local search start from an initial solution and attempt to replace the current solution with a better solution from a defined neighborhood through iterations. A neighborhood is formally defined as follows [5]:

Definition 2.2.
A neighborhood structure is a function $N : S \rightarrow 2^S$ that assigns to every $s \in S$ a set of neighbors $N(s) \subseteq S$. $N(s)$ is called the neighborhood of $s$. Often, neighborhood structures are implicitly defined by specifying the changes that must be applied to a solution $s$ to generate all of its neighbors. The application of such an operator, which produces a neighbor $s \in N(s)$ of a solution $s$, is commonly called a move.

The introduction of a neighborhood structure enables us to define locally minimal solutions.

Definition 2.3.
A locally minimal solution (or local minimum) with respect to a neighborhood structure $N$ is a solution $s^*$ such that $\forall s \in N(s^*) : f(s^*) \leq f(s)$. We call $s^*$ a strict locally minimal solution if $f(s^*) < f(s) \forall s \in N(s^*)$.

The definition of combinatorial optimization problems and a local minimum is important for the taxonomy and the general overview of combinatorial optimization techniques. Before considering approaches to problem-solving techniques, it is necessary to give an overview of problem types (classes) from the aspects of computational complexity and to identify a problems difficulty and its affiliation to a certain class of problems. In combination with that information, it is even more important to define the computational complexity terms. Therefore, topics from computational complexity theory will be covered in the next section.

2.2 Computational Complexity

In an earlier section of this paper, we wrote about the problems (tasks) that can be solved using algorithms. Additionally, there are problems that cannot be solved using any algorithm. They are generalized problems that cannot be solved (undecidable problems) when using modern computers that are based on the principles defined by von Neumann. Because of this fact, even a significant improvement in the computer features will not contribute to solving that type of problem. Among others, the Turing machine halting problem, Hilbert’s tenth problem, the problem of recognition of equivalent words in an associative calculus and others [3] belong to this group of problems.

For us, a much more interesting class of problems is the class that can be solved using algorithms because matching algorithms with different complexities for solving them already exist. Complexity implies the computing resources, which is the processor time and the amount of memory space that are necessary for finding a solution. In practice, the most frequent crucial resource is the processor time. Therefore, depending on the necessary processor time for solving the problem, the algorithms will be divided into classes of complexity.

According to Definition 1, every CO problem is determined by a list of parameters (variables), which represents the input data and the conditions that must be satisfied by the solution. The values of the parameters define the dimension of a problem. For example, if the problem is language recognition, it is usually the length of the words that must be accepted or rejected. If a problem has many parameters, then coding can be accomplished in such a way that every individual task is assigned a dimension in a unique
we can identify problems in NP that are, in a certain sense, the hardest. These problems are called
NP-complete. They have the following property: if there is a polynomial algorithm for one of them, then
there is a polynomial algorithm for any problem in the NP class. Consequently, finding the answer to the
question of whether P = NP can be focused on finding a polynomial algorithm for any of the NP-complete problems.

Most of the results in the resolution of the P and NP problems have been obtained by reducing one problem to another. The transformation of one problem to another is called a reduction. A reduction $f$ is solvable in polynomial time if there is a constant $k$ and algorithm $T$ that for $n$ (the argument length) produces a solution of $f$ in the time $O(n^k)$. A problem $A$ is reducible in polynomial time to a problem $B$ if $A$ reduces to $B$ through a function $f$ that is solvable in polynomial time, and this relationship is denoted as $A \leq_p B$. Reducing in polynomial time is presented using the theorem of reducing [10].

**Theorem 2.1.**

1. If $A \leq_p B$ and $B$ is in P, then $A$ is also in P.
2. If $A \leq_p B$ and $A$ is in NP, then $B$ is also in NP.
3. If $A \leq_p B$ and $B \leq_p C$, then $A \leq_p C$.

If the problem $B$ belongs to the class P and the problem $A$ can be reduced in polynomial time to $B$, then the problem $A$ also belongs to the class P. Additionally, if problem $A$ belongs to the class NP and this problem can be reduced in polynomial time to problem $B$, then problem $B$ also belongs to the class NP. Additionally in part c, if $p, q$ are two functions that have polynomial growth, then their composition $p(q(n))$ also has polynomial growth.

It is considered that problem $A$ is NP-complete if

- it belongs to the class NP,
- every other problem $B$ from NP is in polynomial time reducible to a problem $A$, which is referred to as $B \leq_p A$.

This definition, because of the previous theorem, means the following:

- If $A$ is NP-complete and $A$ is in P, then $P = NP$.
- If $A$ is NP-complete, $B$ is in NP and $A \leq_p B$, then $B$ is also NP-complete.

The initial problem that was directly proved to be NP-complete is the Satisfiability problem of the Boolean formula in a conjunctive normal form (SAT), which is also known as Cook-Levins theorem. Cook has proved that the satisfiability problem is NP-complete. A standard method for proving NP-completeness is to take a known NP-complete problem and reduce it in polynomial time to the problem of NP that we want to prove to be NP-complete. After it is established that SAT is NP-complete, then SAT is reduced to many other problems from NP, and it is proved that the other problems are also NP-complete. For many problems, it has been shown that they represent NP-complete problems. A problem is NP-hard if and only if there is an equivalent decision problem that is NP-complete [11].

When a dimension of an NP-hard problem is large enough, then a technique called heuristics is used for solving the problem. In practice, heuristics are very often used for reaching an optimal solution, although its optimality cannot be verified. The most common verification of a solutions optimality is far more complicated than the solution itself. Heuristics are very popular and are often a choice for solving NP-hard problems, first because of their efficiency and also for the possibility of applying them to problems that have large dimensions.

After we have defined the problem of combinatorial optimization and the terms and concepts of the computational complexity, in the next section, we will focus in detail on the techniques that are used for solving CO problems and on the hybridization strategies.
2.3 Problem-solving Techniques of Combinatorial Optimization

It is worthwhile mentioning that the theoretical definition of efficiently solvable does not necessarily mean unambiguously efficient solvability in practice. Nevertheless, we repeat that for some NP-hard problems that have a small dimension, the exact techniques can be used for solving them; however, in most cases, they give pure results when applied to NP hard problems that have large dimensions.

2.3.1 Exact Techniques

From a methodological point of view, the simplest exact approach would probably be the full enumeration of all of the possible solutions of task S (which is called a brute-force search). Because of the inherent combinatorial explosion that occurs with the increase in dimensions, this approach is only possible, as stated earlier, for problems that have small dimensions. Therefore, all of the practical approaches for exact solutions should search more space, but only indirectly, excluding those spaces in which it is guaranteed that a better solution than the current solution cannot be found. Those techniques are often based on a tree search, where the space of the search is recursively divided by the divide-and-conquer principle into separate divided subspaces, by fixing certain variables or by imposing additional restrictions. The scalability of a tree search depends on the effectiveness of the pruning mechanism. In the branch-and-bound mechanism, the upper and lower bounds are specified for the target values of a solution, and the subspace for which the lower bounds exceed the upper bounds is rejected [7].

Among the many exact methods, the most often-used methods include the branch-and-bound, the branch-and-cut, linear programming, and dynamic programming. It is known that exact methods are typically time-consuming; thus, they cannot be applied to large NP-hard problems. Heuristic techniques are used for solving these problems.

2.3.2 Heuristic Techniques

Heuristic techniques for solving CO problems cover the range from simple constructive techniques, such as ad-hoc greedy algorithms and local search, to a variety of metaheuristic techniques.

Usually, a heuristic is a technique that is used for finding a good solution for a task in a relatively short time, without a guarantee of its admissibility and optimality and often not even a guarantee of its closeness to the optimal solution [8]. Considering the optimality of a solution, i.e., its closeness to the optimal solution, heuristics can be divided into those that guarantee a certain quality of solutions and those that do not give any guarantee.

Approximation algorithms

Approximation algorithms are a special type of heuristics that, unlike all of the other methods, guarantee a certain quality to an approximate solution. Therefore, we can say that giving a sub-optimal solution in polynomial time is a feature of approximation algorithms as well as a guarantee that a solution is close to the optimal. Although it is difficult to find approximation algorithms that have good guarantees for the quality of a solution, the field of approximation algorithms is important because it brings mathematical rigor to the study of heuristics. That circumstance gives us an opportunity to prove how well the heuristic performs on all of the instances of a problem or on only some types of instances [2].

In the following section, according to [7, 11], the solution value of an (arbitrary) instance $I$ of an optimization problem $P$ by algorithm $A$ will be denoted by $A(I)$, whereas the optimal solution value is denoted as $Opt(I)$. 
Definition 2.5.
An approximation algorithm $A$ has an absolute performance guarantee of $k (k > 0)$ if, for every instance $I$, it holds that $|Opt(I) - A(I)| \leq k$.

Absolute approximation algorithms exist for only a few problems; the following algorithms are more common and provide a relative performance guarantee:

Definition 2.6.
An approximation algorithm $A$ for a minimization problem has a relative performance guarantee (which is often called the approximation ratio or approximation factor) $k (k > 1)$ if, for every instance $I$, it holds that $A(I) \leq k \cdot Opt(I)$.

The algorithm $A$ is then also called a $k$-approximation algorithm [7, 11] and a $k$ relative performance guarantee. In this paper, we will follow the convention that $k > 1$ for minimization problems, while $k < 1$ for maximization problems. Thus, a 1/2-approximation algorithm for a maximization problem is a polynomial-time algorithm that always returns a solution whose value is at least half the optimal value.

We might also consider the relative deviation for a minimization problem:

$$\frac{A(I) - Opt(I)}{Opt(I)} \leq \varepsilon \iff A(I) \leq (1 + \varepsilon)Opt(I) \quad (1)$$

An analogous relative deviation for a minimization problem (1), a maximization problem seeks to maximize the value of an approximation algorithm:

$$\frac{Opt(I) - A(I)}{Opt(I)} \leq \varepsilon \iff A(I) \geq (1 - \varepsilon)Opt(I) \quad (2)$$

An approximation algorithm $A$ for a minimization problem with a relative deviation $\varepsilon$ is a $(1 + \varepsilon)$-approximation algorithm. The case for a maximization problem is analogous, and we obtain a $(1 - \varepsilon)$-approximation algorithm.

A richer class of algorithms, in fact sets of algorithms, is when we consider the deviation itself to be an input:

Definition 2.7.
An approximation scheme $A$ is a family of algorithms $\{A_\varepsilon\}$ in which there is an algorithm for each $\varepsilon > 0$ such that $A_\varepsilon$ is a $(1 + \varepsilon)$-approximation algorithm (for minimization problems) or a $(1 - \varepsilon)$-approximation algorithm (for maximization problems).

We can distinguish the approximation schemes according to their runtimes:

Definition 2.8.
An approximation scheme $A$ is a polynomial-time approximation scheme (PTAS) when the runtime of $A$ is polynomial in the size of the instance.

Polynomial-time approximation schemes are the most desirable and flexible type of relative approximation; when selecting parameters in the algorithm, the algorithm can achieve a desired accuracy.

Classical heuristics
The results obtained using classical heuristic techniques (heuristic algorithms for solving the problem at hand) are unreliable, but these techniques can solve NP-hard problems that have large dimensions. Heuristics emphasize having a relatively short period of time searching for a solution; in most cases, when the dimension of an NP-hard problem is large, it is impossible to obtain an exact solution in a reasonable
amount of time. These techniques are used even when it is impossible to define a mathematical model for
the problems that have no nice features, such as convexity, differentiability, and other attributes. It can be
experimentally concluded whether such a technique is better than any other previously known techniques.

Classical heuristics are developed with the aim of solving problems at hand, and they are bounded
by the characteristics of that problem. Progress made in the areas of data structures, organization and
methods of storing data, as well as in improving the features of the computers themselves, has led to the
reprogramming of classical heuristics. In this way, researchers have created a new class of heuristics, the
so-called modern heuristics or metaheuristics. Unlike classical heuristics, metaheuristics contain rules and
principles that can be applied to solve a large number of practical problems in various areas. As will be
discussed in more detail below, each metaheuristic is a general technique and can be successfully applied
to a broad class of problems.

Metaheuristics Over the past 30 years, a new type of algorithm has been developed, which in essence
combines classical heuristic techniques with a conceptually higher level for the purpose of efficiently
exploring the space of potential solutions. These techniques are now typically referred to as metaheuristics.
The term metaheuristics was first introduced by Glover in 1986. Before accepting this term, the term
modern heuristics was often used [5]. The origins of metaheuristics are to be found in the Artificial
Intelligence and Operations Research field [4].

Many authors have given their definition of a metaheuristic. Osman [12] has defined metaheuristic as a
main iterative process that guides and modifies the operations of subordinate heuristics with the aim of
efficiently creating a high-quality solution. In addition, metaheuristics can intelligently combine different
concepts about the search space of potential solutions using adaptive strategies and structured information.
Blum and Roli [5] summarize the different definitions of the term and state the fundamental properties
that characterize metaheuristics:

- metaheuristics are strategies that guide the search process,
- the objective is to efficiently explore a search space to find the (near-) optimal solutions,
- techniques that constitute metaheuristic algorithms range from simple local search procedures to
  complex learning processes,
- the algorithms that they use are approximate and usually nondeterministic algorithms,
- they use mechanisms to avoid falling into the local optima of the search space,
- basic concepts permit an abstract level description,
- they are not oriented to a specific problem,
- they can use domain-specific knowledge in the form of heuristics that are controlled by an upper-
  level strategy,
- they use search experience to guide the search.

We note that the main characteristic of the metaheuristic techniques, with some modification, is the
possibility of being applied to a wide range of combinatorial optimization problems because they are not
oriented to a specific problem. These techniques search the set of feasible solutions for the purpose of
finding the best possible solution with the moves that are allowed: (1) moving toward a worse solution
compared with the current solution, (2) expanding the search space with unacceptable elements, (3)
searching for a solution that combines existing solutions, among other moves. Metaheuristics can be
divided into different methods, and the division that distinguishes the metaheuristics according to the
The number of solutions that are used at the same time is usually mentioned. Thus, metaheuristics can be divided into two categories [6]: single solution algorithms and population-based algorithms.

Typical representatives of metaheuristic techniques are the following:

- Genetic algorithms (GA),
- Particle swarm optimization (PSO),
- Ant colony optimization (ACO),
- Simulated annealing (SA),
- Tabu search (TS),
- Greedy Randomized Adaptive Search Procedure (GRASP),
- Variable neighborhood search (VNS) and
- various other metaheuristic techniques.

Each of these metaheuristic techniques has its own background. Some of the techniques are inspired by natural processes such as evolution; others are an extension of some of the less sophisticated algorithms, such as greedy algorithms and local search algorithms. When it became clear that clean metaheuristics have reached their limits, researchers turned to a combination of different algorithms. In recent years, a pretty impressive number of algorithms no longer exclusively follow a paradigm of traditional metaheuristics. Instead, they combine the various algorithmic components, whose roots often originate from algorithms that belong to different areas of optimization. These algorithms are called hybrid metaheuristics [4].

### 2.3.3 Hybrid Metaheuristics

The main motivation for combining different techniques is to take advantage of the complementary character of different optimization algorithms through various hybridization strategies; in other words, the hybrids are likely to benefit from synergies. Akkoc [13] states that the advantage of hybrids is that each technique brings its own advantages. Recently, many papers [14–17] have sought to exploit the complementarities of different techniques. It has been determined that the selection of an adequate complementary combination of techniques and algorithmic concepts can be the key to achieving high performance in solving many difficult optimization problems. Unfortunately, this problem is usually a difficult task that requires knowledge of different areas of optimization. In this context, further progress can be expected through the development of new features in framework tools that have emerged recently.

In the literature, hybridization and cooperation are often used in the same sense and indicate processes that combine a variety of optimization techniques. Initially, hybridization was mainly implemented among multiple metaheuristics. Currently, collaboration between metaheuristics and other techniques has been increasingly proposed [6]. Such a collaboration strategy usually gives better results because it can take advantage of more types of techniques simultaneously. Blum et al. [4] stated that metaheuristics can be hybridized with the following:

- other metaheuristics,
- constraint programming techniques,
- tree search techniques,
Hybrid Techniques of Combinatorial Optimization with Application to Retail Credit Risk Assessment

Figure 1. The four strategies according to the taxonomy of hybrid metaheuristics [6]

- problem relaxation techniques and
- dynamic programming techniques.

The cooperation of different techniques in the design of hybrid metaheuristics can be viewed from different perspectives [6]. According to the level at which it takes place, cooperation can be at the following:

- Low level: The result is always one functional optimization technique that is composed. Certain functions within metaheuristics are replaced or supplemented by other techniques.
- High level: Different algorithms are self-sufficient.

According to the sequence of operations, cooperation can entail the following:

- Relay operation: A set of techniques is applied one after the other, each using the previous output as its input and acting as a pipeline.
- Teamwork: Represents cooperative optimization models.

Figure 1 shows the four classes of hybrid metaheuristics that we obtain as results when we combine (1) the level at which hybridization occurs and (2) the way in which hybridization is performed with respect to the sequence of operations. Perhaps even more importantly, this categorization, apart from showing the results of a combination, also show hybridization approaches or hybridization strategies or cooperation schemes. This arrangement shows the possible ways of creating new hybrid techniques.

**LR strategy (low-level relay)**

This strategy corresponds to a combination in which a certain technique is embedded in another technique. The built-in technique (or techniques) is executed sequentially, and the execution of the global technique depends on the results that are obtained when using the embedded technique. This type of collaboration is common when the metaheuristic technique is used to improve the speed of an exact technique (Figure 2). In the context of cooperation between the metaheuristics, as part of this strategy, the following type of cooperation is usually implemented. First, it initiates an evolutionary algorithm, which then initiates a local search, to intensify the search for the best solutions area [20]. An evolutionary
algorithm directs the search for potential solutions to unvisited areas, while a local search explores the neighborhoods of the best solutions.

In the context of cooperation between metaheuristic and exact techniques, the following approach can be considered: exact techniques provide potential solutions that are used to define the search space for the metaheuristic technique. Then, the results that are obtained by the metaheuristic technique can be analyzed by a different algorithm. A hybridization of the genetic algorithm with some exact techniques belongs to this strategy, and it is used in an HGA-NN metaheuristic [18]. In the experimental part of this study, we will use this strategy.

**LT strategy (low-level teamwork)**

In this cooperation scheme, part of one technique is replaced or supplemented with one or several other techniques. This type of collaboration can drastically improve the metaheuristics. A strategy of this class incorporates a technique (techniques) that can be executed in parallel with the global technique. A classic example of the LT strategy is a memetic algorithm that combines a genetic algorithm and local search techniques, which replace the transformation operator in the genetic algorithm, usually a mutation.

A significant improvement of a metaheuristic that is in the LT class is shown in the paper in [19]. A cooperation scheme is based on the technique of local search and an adaptive procedure that uses complex structures in the neighborhood to choose the neighboring solution that will be analyzed in the next step. For this purpose, in the heuristics, there are built-in techniques for mixed integer programming. In [20], Cotta et al. presented a hybridization strategy that combines the strengths of two very different techniques, a genetic algorithm and the branch-and-bound technique. Because of the diversity of these techniques, a new technique has been developed in such a way that each technique represents a tool of the other technique (Figure 3).

**HR strategy (high-level relay)** In the HR hybridization strategy, different techniques are self-sufficient and are performed in a sequence. This cooperation scheme is the most common for general hybridization. Similar to in other schemes of cooperation, different combinations for solving problems could exist. However, in general, the most natural approach is to design a sequential execution in which the metaheuristic is launched before the exact techniques, to enable the metaheuristic to provide results of the exact technique (Figure 4). These results help the exact technique of search to define the search space and to reduce it before starting the search. In the above example, the search space could be reduced to the neighborhoods around the proposed solutions.

An example of this type of hybridization strategy is proposed by Bent and Van Hentenryck [21] in their two-stage algorithm for the transportation problem. This algorithm first minimizes the number of vehicles using the technique of simulated annealing. Afterward, it minimizes the travel costs using the technique

---

**Figure 2.** Diagram in which a metaheuristic technique searches a wider area of solutions and optimizes the exact technique.
Figure 3. Diagram of the hybrid genetic algorithm [20]

Figure 4. Metaheuristic provides some initial solutions / borders

of large neighborhood search.

**HT strategy (high-level teamwork)**

This strategy contains combinations in which independent search techniques are performed in a parallel and cooperative manner. Cooperation between metaheuristics as well as metaheuristics and exact techniques is conducted as though it is between parallel islands. The only difference is that there are two different types of islands: some are devoted to an exact search while others perform a heuristic search. During the execution, different algorithms exchange information (Figure 5), and the main difficulty is to determine when and how to exchange information. This problem tends to be solved by a parameter.

The paper in [22] combines simulated annealing and the branch-and-bound technique as an example of this type of hybridization strategy. Parallel computers are used to simultaneously perform the simulated annealing technique and the branch-and-bound technique, which mutually exchange information to influence the flow and the results of the search. The reported results for the broader spectrum of mixed integer programming models have shown the efficiency of using this hybridization strategy.

There is a very thin line between the mentioned hybridization strategies. However, the purpose is achieved because the shown strategies can serve as a guide and a source of ideas for creating new hybrid techniques, but also they can serve as a way of analyzing and systematizing the existing hybrid techniques. It should be noted that when low-level strategies are used and low-level techniques are created, it is necessary to intervene in the implementation of the technique, while the HR strategy mostly implies an association from the outside of two or more independent techniques. We are also aware that there is no division that could accurately systematize all of the hybrid techniques; therefore, there is a variety of
2.4 Diversification and Intensification

We have started with a brief overview of the techniques for solving the problems of combinatorial optimization, in which the basic types of techniques, the strategies of combining them and the potential benefits of such a combination are presented. In this section, we will show that the fundamental concepts and principles in solving combinatorial optimization problems must address hybrid metaheuristics.

A hybrid metaheuristic can be seen as a high-level technique that searches the space of solutions using a strategy. It is of great importance to establish a dynamic balance between the diversification and intensification. Diversification implies a force that directs the search for potential solutions to unvisited areas, thereby generating solutions that are in many ways different from the previously seen solutions. Intensifying focuses the search on exploring the neighborhoods of the best solutions that have been found thus far [23].

The balance between the diversification and intensification is important for the rapid identification of areas in the search space that have high-quality solutions without losing too much time on the areas in the search space that have already been explored or the areas that do not give good solutions. There are different techniques for achieving this balance. Some of these techniques are highly intelligent solutions and belong to a group of local search techniques [5]. Their aim is to avoid the local minima and to continue searching the solution space in the hope of finding a better local minimum or the global minimum. This group includes techniques such as simulated annealing, tabu search, greedy randomized adaptive search procedure, a variable neighborhood search and other similar techniques.

The second major group consists of the techniques that have a learning component in their philosophy. They implicitly or explicitly attempt to learn the correlation between the independent and dependent variables and to identify areas of high-quality solutions in the search space. This group includes population-based techniques, such as genetic algorithms, particle swarm optimization, ant colony optimization, and other similar techniques.

Certain basic techniques that build hybrid techniques have different strategies for solving the problem of diversification and intensification, and in these differences lies the space for creating high-quality hybrid techniques. Hybrid techniques can combine different strategies of basic techniques and overcome some weaknesses of an individual technique, or better yet, take advantage of both through synergism. A targeted approach of the development of hybrid metaheuristics can achieve, in addition to the benefits, an unintended feature that manifests itself in the fact that the solution (a new hybrid) is strictly focused on possible combinations in the presented strategy.

Figure 5. Two algorithms work in parallel and exchange information
A solution search strategy could be described as follows: attractive areas should be searched more thoroughly compared to less attractive areas. Therefore, many of the techniques that were previously surveyed remember the quality of the solutions, to enable them not only to ultimately determine the best solution at the end of search but also to explore the neighborhood of better solutions more thoroughly during the search.

In Figure 6, the areas of better solutions, the areas of local and global minima, are marked by ellipses. Figure 7 shows, using points, the intensity of the search in specific areas. Figure 6 and Figure 7 show one of the main challenges during the search of the solution space: getting out of the field of local minima to the global minimum of the target function. Therefore, when considering the issue of diversification and
intensification, Glover and Laguna [23] suggest that the whole process takes place in the strategic form of oscillation around these concepts.

The balance between the diversification and intensification has been in the focus of many recent studies in the field of CO. We will name some typical representatives of papers that demonstrate such a balance. The paper of Akpinar et al. [25] shows a hybrid algorithm that combines ant colony optimization with a genetic algorithm. Rama Mohan Rao and Shyju [26] combine the features of simulated annealing and tabu search to present the effectiveness of combining the good features of these metaheuristics. To maintain a balance between intensification and diversification, they propose an algorithm that has multiple origins. Mashinchi et al. [27] proposed a hybrid that was based on tabu search and the Nelder-Mead search strategies. Tabu search is responsible for the diversification, and the Nelder-Mead strategy focuses on intensifying the search. Papers [28–30] also provide examples of achieving a balance of these concepts.

In fact, at a deeper level, the balance between these two concepts is the focal point of all of the hybrid techniques of CO. In some way, all of the hybridization strategies are looking for a balance between diversification and intensification, which are the key concepts of hybridization. Furthermore, heuristic algorithms for solving CO problems are looking for the same balance. We can say that diversification and intensification are the focal points not only of the hybrid techniques of CO but also in the solution search strategies of NP-hard problems, in general.

In the following section, the hybrid technique of combinatorial optimization for feature selection and classification with an application to a German credit dataset will be shown.

### 3. EMPIRICAL ANALYSIS OF HYBRIDIZATION STRATEGIES

Below we present one example of the low-level relay strategy, which is one of the previously described strategies of combining different techniques in a hybrid technique. The hybrid technique that is shown is called HGA-NN and is composed of fast filter techniques, a hybrid genetic algorithm (HGA) and an artificial neural network (NN). Hence, this technique combines fast feature selection techniques with the main technique, the genetic algorithm, to make an efficient hybrid technique. The built-in fast filter techniques can be executed in parallel, but they are executed before with regard to the global technique, and the execution of the global technique depends on the results that are obtained using the embedded techniques. The result is that one functional optimization technique is composed. The function of the initial population generation within the genetic algorithm is supplemented by the results of fast filter techniques. This arrangement is the main reason why we say that the genetic algorithm is hybridized. Research was conducted on solving the problems of feature selection and classification when assessing retail credit risks on the German credit dataset.

#### 3.1 Model Description

Each hybridization technique is based on certain assumptions. Thus, this technique is based on the basic assumption that if a priori information about the potentially attractive areas of a solution is available, then the initial population of the GA can be generated in such a way that it uses this information and reduces the dimensionality of the problem to those features that form attractive areas. The detailed description can be found in [18]. The results of earlier experience and the results of the fast filter techniques are considered to be a priori information about the attractive areas. Therefore, we combine fast feature selection techniques with a genetic algorithm. In fact, a priori information comprises initial solutions that are included in the initial population of the GA, and the remainder of the initial population is filled
hybrid techniques of combinatorial optimization with application to retail credit risk assessment

Figure 8. The proposed hybrid technique

randomly. In the next step, the current solution is regarded as the a priori information and the new piece of information is utilized to update our mathematical model, i.e., the incremental improvement can be seen as Bayesian updating [31]. The proposed algorithm is shown in Figure 8.

The hybrid technique of CO, which is shown in Figure 8, uses fast filter techniques (Information gain, Gain ratio, Gini index and Correlation) to restrict the search space for the metaheuristic technique, i.e., for the genetic algorithm. Additionally, fast exact techniques provide potential solutions that are included in the initial population of the GA. The described procedure directs the search to the attractive areas, i.e., it performs intensification, while the balance between the diversification and the intensification is achieved thru the native operators that are included in the genetic algorithm.

3.2 Description of the Dataset

Tsai and Cheng [32] indicate that the German credit dataset is a challenging benchmark for bankruptcy prediction. Therefore, to reliably and effectively examine the performance of the prediction models, one should at least consider the German dataset to be a benchmark for evaluation. Accordingly, the credit dataset that is used in this experiment is the German credit dataset. This dataset is available from the UCI Repository of Machine Learning Databases [33], and it is composed of 700 instances of creditworthy applicants and 300 instances of bad credit applicants. The original dataset, in the form provided by Professor Dr. Hans Hofmann of the University of Hamburg, contains, for each applicant, 20 input features. This original dataset had 13 categorical features, some of which have been transformed into a series of binary features in such a way that they can be appropriately handled by the NN. Several ordered categorical features have been left as is and are treated as numerical. The transformed German credit dataset contains 30 regular features of the integer data type and two (id, label) special features, and it can be reached at <http://ocw.mit.edu/courses/sloan-school-of-management/15-062-data-mining-spring-2003/download-course-materials/>. All of the features that have descriptive statistics and that have a mark entered for the reduced set of features are shown in Table 1.
ARTIFICIAL INTELLIGENCE AND APPLICATIONS

Table 1. Transformed German credit dataset with descriptive statistics and with a mark for being entered into the reduced set

<table>
<thead>
<tr>
<th>Attr.</th>
<th>Code</th>
<th>Description</th>
<th>Statistics</th>
<th>Range</th>
<th>Reduced subset</th>
</tr>
</thead>
<tbody>
<tr>
<td>att1</td>
<td>ID</td>
<td>Observation No.</td>
<td>avg = 500.500 +/- 288.819</td>
<td>[1; 1000]</td>
<td>✓</td>
</tr>
<tr>
<td>att2</td>
<td>CHK_ACCT</td>
<td>Checking account status</td>
<td>avg = 1.577 +/- 1.258</td>
<td>[0; 3]</td>
<td>✓</td>
</tr>
<tr>
<td>att3</td>
<td>DURATION</td>
<td>Duration of credit in months</td>
<td>avg = 20.903 +/- 12.059</td>
<td>[4; 72]</td>
<td>✓</td>
</tr>
<tr>
<td>att4</td>
<td>HISTORY</td>
<td>Credit history</td>
<td>avg = 2.545 +/- 1.083</td>
<td>[0; 4]</td>
<td>✓</td>
</tr>
<tr>
<td>att5</td>
<td>NEW_CAR</td>
<td>Purpose of credit</td>
<td>avg = 0.234 +/- 0.424</td>
<td>[0; 1]</td>
<td>✓</td>
</tr>
<tr>
<td>att6</td>
<td>USED_CAR</td>
<td>Purpose of credit</td>
<td>avg = 0.103 +/- 0.304</td>
<td>[0; 1]</td>
<td>✓</td>
</tr>
<tr>
<td>att7</td>
<td>FURNITURE</td>
<td>Purpose of credit</td>
<td>avg = 0.181 +/- 0.385</td>
<td>[0; 1]</td>
<td>✓</td>
</tr>
<tr>
<td>att8</td>
<td>RADIO/TV</td>
<td>Purpose of credit</td>
<td>avg = 0.280 +/- 0.449</td>
<td>[0; 1]</td>
<td>✓</td>
</tr>
<tr>
<td>att9</td>
<td>EDUCATION</td>
<td>Purpose of credit</td>
<td>avg = 0.050 +/- 0.218</td>
<td>[0; 1]</td>
<td>✓</td>
</tr>
<tr>
<td>att10</td>
<td>RETRAINING</td>
<td>Purpose of credit</td>
<td>avg = 0.097 +/- 0.296</td>
<td>[0; 1]</td>
<td>✓</td>
</tr>
<tr>
<td>att11</td>
<td>AMOUNT</td>
<td>Credit amount</td>
<td>avg = 3271.258 +/- 2822.737</td>
<td>(250; 18424)</td>
<td>✓</td>
</tr>
<tr>
<td>att12</td>
<td>SAV_ACCT</td>
<td>Average balance in savings account</td>
<td>avg = 1.105 +/- 1.580</td>
<td>[0; 4]</td>
<td>✓</td>
</tr>
<tr>
<td>att13</td>
<td>EMPLOYMENT</td>
<td>Present employment since</td>
<td>avg = 2.384 +/- 1.208</td>
<td>[0; 4]</td>
<td>✓</td>
</tr>
<tr>
<td>att14</td>
<td>INSTALL_RATE</td>
<td>Installment rate as % of disposable income</td>
<td>avg = 2.973 +/- 1.119</td>
<td>[1; 4]</td>
<td>✓</td>
</tr>
<tr>
<td>att15</td>
<td>MALE_DIV</td>
<td>Applicant is male and divorced</td>
<td>avg = 0.050 +/- 0.218</td>
<td>[0; 1]</td>
<td>✓</td>
</tr>
<tr>
<td>att16</td>
<td>MALE_SINGLE</td>
<td>Applicant is male and single</td>
<td>avg = 0.548 +/- 0.498</td>
<td>[0; 1]</td>
<td>✓</td>
</tr>
<tr>
<td>att17</td>
<td>MALE_MAR_WID</td>
<td>Applicant is male and married or a widower</td>
<td>avg = 0.092 +/- 0.289</td>
<td>[0; 1]</td>
<td>✓</td>
</tr>
<tr>
<td>att18</td>
<td>CO-APPLICANT</td>
<td>Application has a co-applicant</td>
<td>avg = 0.041 +/- 0.198</td>
<td>[0; 1]</td>
<td>✓</td>
</tr>
<tr>
<td>att19</td>
<td>GUARANTOR</td>
<td>Applicant has a guarantor</td>
<td>avg = 0.052 +/- 0.222</td>
<td>[0; 1]</td>
<td>✓</td>
</tr>
<tr>
<td>att20</td>
<td>PRESENT_RESIDENT</td>
<td>Present resident since - years</td>
<td>avg = 2.845 +/- 1.104</td>
<td>[1; 4]</td>
<td>✓</td>
</tr>
<tr>
<td>att21</td>
<td>REAL_ESTATE</td>
<td>Applicant owns real estate</td>
<td>avg = 0.282 +/- 0.450</td>
<td>[0; 1]</td>
<td>✓</td>
</tr>
<tr>
<td>att22</td>
<td>PROP_UNKN</td>
<td>Applicant owns no property (or unknown)</td>
<td>avg = 0.154 +/- 0.361</td>
<td>[0; 1]</td>
<td>✓</td>
</tr>
<tr>
<td>att23</td>
<td>AGE</td>
<td>Age in years</td>
<td>avg = 35.546 +/- 11.375</td>
<td>[19; 75]</td>
<td>✓</td>
</tr>
<tr>
<td>att24</td>
<td>OTHER_INSTALL</td>
<td>Applicant has other installment plan credit</td>
<td>avg = 0.186 +/- 0.389</td>
<td>[0; 1]</td>
<td>✓</td>
</tr>
<tr>
<td>att25</td>
<td>RENT</td>
<td>Applicant rents</td>
<td>avg = 0.179 +/- 0.384</td>
<td>[0; 1]</td>
<td>✓</td>
</tr>
<tr>
<td>att26</td>
<td>OWN_RES</td>
<td>Applicant owns residence</td>
<td>avg = 0.713 +/- 0.453</td>
<td>[0; 1]</td>
<td>✓</td>
</tr>
<tr>
<td>att27</td>
<td>NUM_CREDITS</td>
<td>Number of existing credits at this bank</td>
<td>avg = 1.407 +/- 0.578</td>
<td>[1; 4]</td>
<td>✓</td>
</tr>
<tr>
<td>att28</td>
<td>JOB</td>
<td>Nature of job</td>
<td>avg = 1.904 +/- 0.654</td>
<td>[0; 3]</td>
<td>✓</td>
</tr>
<tr>
<td>att29</td>
<td>NUM_DEPENDENTS</td>
<td>Number of people for whom liable to provide maintenance</td>
<td>avg = 1.155 +/- 0.362</td>
<td>[1; 2]</td>
<td>✓</td>
</tr>
<tr>
<td>att30</td>
<td>TELEPHONE</td>
<td>Applicant has phone in his or her name</td>
<td>avg = 0.404 +/- 0.491</td>
<td>[0; 1]</td>
<td>✓</td>
</tr>
<tr>
<td>att31</td>
<td>FOREIGN</td>
<td>Foreign worker</td>
<td>avg = 0.037 +/- 0.189</td>
<td>[0; 1]</td>
<td>✓</td>
</tr>
<tr>
<td>att32</td>
<td>LABEL</td>
<td>Credit rating</td>
<td>mode = 1 (700), least = 0 (300)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.3 Experimental Results and Discussion

The initial parameters of the genetic algorithm were set according to a comprehensive suite of experiments that were carefully designed. From one execution to another, the parameters did not change except for the parameters for the selection scheme and crossover type, which are highlighted in each row of Table 2.

According to these, we made measurements, which are expressed in terms of the percentage of the
Hybrid Techniques of Combinatorial Optimization with Application to Retail Credit Risk Assessment

Table 2. The prediction accuracy expressed as a %

<table>
<thead>
<tr>
<th>Execution</th>
<th>Selection scheme</th>
<th>Crossover type</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>5</th>
<th>8</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Roulette wheel</td>
<td>uniform</td>
<td>77.8</td>
<td>77.8</td>
<td>77.8</td>
<td>77.8</td>
<td>78.1</td>
<td>78.3</td>
<td>78.3</td>
<td>78.3</td>
<td>78.3</td>
</tr>
<tr>
<td>2.</td>
<td>Tournament</td>
<td>uniform</td>
<td>77.8</td>
<td>77.8</td>
<td>78.4</td>
<td>78.4</td>
<td>78.4</td>
<td>78.6</td>
<td>78.6</td>
<td>79.0</td>
<td>79.0</td>
</tr>
<tr>
<td>3.</td>
<td>Tournament</td>
<td>uniform</td>
<td>77.8</td>
<td>78.0</td>
<td>78.0</td>
<td>78.7</td>
<td>78.7</td>
<td>78.7</td>
<td>79.0</td>
<td>79.0</td>
<td>79.0</td>
</tr>
<tr>
<td>4.</td>
<td>Stochastic</td>
<td>one point</td>
<td>77.8</td>
<td>78.5</td>
<td>78.6</td>
<td>78.6</td>
<td>78.7</td>
<td>78.7</td>
<td>78.7</td>
<td>78.8</td>
<td>78.8</td>
</tr>
<tr>
<td>5.</td>
<td>Boltzmann</td>
<td>uniform</td>
<td>77.8</td>
<td>77.8</td>
<td>77.8</td>
<td>77.8</td>
<td>78.4</td>
<td>78.4</td>
<td>78.4</td>
<td>78.4</td>
<td>78.5</td>
</tr>
<tr>
<td>6.</td>
<td>Cut</td>
<td>uniform</td>
<td>77.8</td>
<td>78.5</td>
<td>78.5</td>
<td>78.5</td>
<td>78.5</td>
<td>78.5</td>
<td>79.0</td>
<td>79.0</td>
<td>79.0</td>
</tr>
<tr>
<td>7.</td>
<td>Unique</td>
<td>uniform</td>
<td>77.8</td>
<td>78.5</td>
<td>78.5</td>
<td>78.5</td>
<td>78.5</td>
<td>78.5</td>
<td>78.8</td>
<td>78.8</td>
<td>78.8</td>
</tr>
<tr>
<td>8.</td>
<td>Unique</td>
<td>one point</td>
<td>77.8</td>
<td>77.8</td>
<td>77.8</td>
<td>78.3</td>
<td>78.3</td>
<td>78.3</td>
<td>79.0</td>
<td>79.0</td>
<td>79.0</td>
</tr>
<tr>
<td>9.</td>
<td>Tournament</td>
<td>uniform</td>
<td>77.0</td>
<td>79.0</td>
<td>79.0</td>
<td>79.0</td>
<td>79.0</td>
<td>79.0</td>
<td>79.0</td>
<td>79.0</td>
<td>79.0</td>
</tr>
<tr>
<td>10.</td>
<td>Tournament</td>
<td>one point</td>
<td>78.0</td>
<td>78.1</td>
<td>78.4</td>
<td>78.4</td>
<td>78.6</td>
<td>78.6</td>
<td>78.6</td>
<td>79.0</td>
<td>79.0</td>
</tr>
<tr>
<td>11.</td>
<td>Cut</td>
<td>one point</td>
<td>77.9</td>
<td>78.0</td>
<td>78.0</td>
<td>78.1</td>
<td>78.5</td>
<td>78.5</td>
<td>78.7</td>
<td>78.8</td>
<td>79.1</td>
</tr>
</tbody>
</table>

Mean: 77.75 | 78.16 | 78.25 | 78.37 | 78.52 | 78.55 | 78.74 | 78.83 | 78.90

Std. dev.: 0.258 | 0.403 | 0.398 | 0.364 | 0.236 | 0.202 | 0.250 | 0.253 | 0.297

* The best solution

The HGA-NN technique in the 8-th execution reached the best average prediction accuracy of 79.4%, with a unique selection scheme and a one-point crossover type. The best solution was achieved with the following 12 features: checking the account status, the duration of credit in months, the credit history, the purpose of the credit (NEW_CAR), the purpose of the credit (RADIO/TV), the credit amount, the average balance in a savings account, the installment rate as a % of disposable income, the applicant is male and single, the applicant has a guarantor, the number of people who are liable to provide maintenance and foreign workers. The experiment results are presented in Table 2.

The results of our HGA-NN technique were compared with results presented in the literature on other techniques on the German credit dataset. The average prediction accuracy is the most commonly used performance measure. The average accuracy and the average error rates shown in Table 3 refer to the results on the validation data set. The comparative overview indicates that the proposed HGA-NN algorithm provides an average prediction accuracy of 78.9%, which is the best average prediction accuracy. This result confirms achieving a good balance between the diversification and intensification inside the algorithm.

In the German credit data set, the ratio of default cases and non-default cases is 30:70. In such an imbalanced classification problem, the detection of rare events is emphasized and quantified through different misclassification costs. The class imbalance and different misclassification costs present significant challenges to the feature selection and classification applications. Many techniques have been proposed to solve this type of binary classification problem through either data [34] or algorithmic approaches [35, 36]. The data approach is usually based on different sampling techniques, including over-sampling of the minority class or under-sampling of the majority class [34, 37]. The algorithmic approach introduces the following: (i) unequal weights for two classes in the training strategy to force the classifier to pay more attention to the minority class, (ii) a change in the threshold of the classifier or (iii) a change in the fitness function. The mentioned techniques are often used to alleviate the adverse impacts of imbalanced data. They improve the performance of the learners when built from imbalanced data.

To precisely estimate the total relative misclassification costs, we need some additional computing that is based on the error rates and relative costs.

The total relative cost of misclassification (RC) can be computed according to the probability of the errors presented in Table 3, the relative cost of the errors presented in Table 4 and formula [42]:
Table 3. The comparison of the results with other techniques

<table>
<thead>
<tr>
<th>Technique</th>
<th>Probability of errors</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PI(%)</td>
<td>PII(%)</td>
</tr>
<tr>
<td>HGA-NN</td>
<td>46.61</td>
<td>10.17</td>
</tr>
<tr>
<td>SVM[38]</td>
<td>37.00</td>
<td>18.00</td>
</tr>
<tr>
<td>LogR[39]</td>
<td>50.66</td>
<td>11.69</td>
</tr>
<tr>
<td>Bagging/MLP[40]</td>
<td>49.40</td>
<td>24.60</td>
</tr>
<tr>
<td>Logit[41]</td>
<td>18.33</td>
<td>44.00</td>
</tr>
</tbody>
</table>

Table 4. Comparison of costs

<table>
<thead>
<tr>
<th>Technique</th>
<th>Cost ratio (C_I : C_{II})</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1:1</td>
</tr>
<tr>
<td>HGA-NN</td>
<td>0.2110*</td>
</tr>
<tr>
<td>SVM[38]</td>
<td>0.2370</td>
</tr>
<tr>
<td>LogR[39]</td>
<td>0.2338</td>
</tr>
<tr>
<td>Bagging/MLP[40]</td>
<td>0.3204</td>
</tr>
<tr>
<td>Logit[41]</td>
<td>0.3630</td>
</tr>
</tbody>
</table>

* The best model for each ratio.

\[ RC = \alpha(P_I C_I) + (1 - \alpha)(P_{II} C_{II}) \]

where \( \alpha \) is the probability of being a bad client, \( P_I \) is the probability of a type I error, \( C_I \) is the relative cost of a type I error, \( P_{II} \) is the probability of a type II error, and \( C_{II} \) is the relative cost of a type II error. The \( RC \) of each model is computed for seven scenarios, while the best model for each scenario is the model that has the lowest \( RC \) value. Table 4 considers different misclassification costs.

In the presented HGA-NN technique, we did not use any activity to mitigate the adverse impacts of imbalanced data and different misclassification costs. We optimized the average prediction accuracy as a fitness function, which is the most common benchmark measure. According to this strategy, the achieved results are the best in terms of the overall accuracy and relative misclassification costs, with ratio of 1:1, but not for all of the other ratios.

The comparative overview that is shown in Table 3 indicates that the proposed HGA-NN algorithm is an acceptable alternative to optimizing both the feature subset and the neural network parameters for the credit risk assessment on the German credit data set and through it, to optimizing the set of network weights that provide the best average accuracy.

4. CONCLUSIONS

The hybrid techniques of combinatorial optimization have been a subject of interest to many researchers in recent years. The main motivation for the research in this field was provided by thousands of phenomena in the real world that can be formulated, on an abstract level, as combinatorial optimization problems. The knowledge of strategic concepts and the quality of certain optimization techniques is a prerequisite for creating a deeper understanding of hybrid techniques.

Accordingly, we have presented a brief but comprehensive overview of the important concepts in the field of combinatorial optimization and computational complexity theory as well as hybridization strategies and the concepts used in solving combinatorial optimization problems. In addition, one hybrid technique
Hybrid Techniques of Combinatorial Optimization with Application to Retail Credit Risk Assessment

for feature selection and classification in credit risk assessment has been presented. A comparison of the results on the German dataset has shown that the presented hybrid technique outperforms the results that were published in the literature. This finding supports the hypothesis that hybridization results in the enhancement of the classifier performance.

The presented technique has optimized the average accuracy as the fitness function. If banks want to optimize the cost function for some of the ratios instead of the accuracy function, the HGA-NN technique probably can solve this request using an algorithmic approach, i.e., by modification of the fitness function. This assumption is based on the fact that inside the HGA-NN technique, a balance is achieved between the diversification and the intensification, which is shown by having the best result with an average accuracy, and a modification in the fitness function will efficiently direct the search to another objective. Experiments to control different error ratios remain as a task for future research, as well as the testing of the potential and usefulness of our technique with respect to other domains.

References


[35] N. V. Chawla, N. Japkowicz, and A. Kotcz, “Editorial: special issue on learning from imbalanced...


