RMIT UNIVERSITY

Structural Optimization Using Evolutionary Multimodal and Bilevel Optimization Techniques

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

Md. Jakirul Islam

B. Sc. Eng. in Computer Science and Engineering, Dhaka University of Engineering and Technology, Gazipur, Bangladesh, M. Eng. Science, University of Malaya, Malaysia

School of Science

College of Science, Engineering and Health

RMIT University

July, 2018

Declaration

I certify that except where due acknowledgement has been made, the work is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program; and, any editorial work, paid or unpaid, carried out by a third party is acknowledged.

Md. Jakirul Islam
School of Computer Science and Information Technology
RMIT University
July, 2018

Acknowledgments

I would like to express my deepest gratitude towards my supervisors, Professor Xiaodong Li and Dr. Yi Mei, for their invaluable guidance and support throughout my PhD work. This work would not have been possible without the encouragement and motivation that my supervisors provided along with their immense technical expertise. I have learnt a lot from them and it has been my honor to work with them.

Beside my supervisors, I would like to thank Professor Kalyanmoy Deb for his motivation and guidance for the completion of my research work. I would also like to thank all the members of ECML group. I have some level of interactions with all of you and had the opportunity to learn many things from you, which I truly appreciate. My sincere thanks also go to RMIT University for offering me valuable financial support under the RPIS scholarship scheme. Without this financial support, my ambition to study abroad would not have been realized.

I would like to thank my family and friends whose unconditional love and support have carried me through the tough time. I must thank my wife for her support and endless patience during my PhD study.

Last but not the least; I would like to thank Almighty Allah who gave me strength and made me capable of completing my PhD research on time.

Credits

Portions of the material in this thesis have previously appeared in the following publications:

- **Md. Jakirul Islam**, Xiaodong Li, and Yi Mei, "A time-varying transfer function for balancing exploration and exploitation ability of a binary PSO". Applied Soft Computing 59: 182 - 196, 2017. Publisher: Elsevier.
- **Md. Jakirul Islam**, Xiaodong Li, and Kalyanmoy Deb, "Multimodal Truss Structure Design Using Bilevel and Niching Based Evolutionary Algorithms". In: *Proceedings of the Genetic and Evolutionary Computation Conference*, pp 274-281, 2017. Publisher: ACM.
- **Md. Jakirul Islam**, Xiaodong Li, and Kalyanmoy Deb, "A Bilevel Niching Method for Solving Truss Design Problems". Submitted to the journal of Applied Soft Computing. Current status: Under review. Publisher: Elsevier .
- **Md. Jakirul Islam**, Xiaodong Li, and Yi Mei, "A framework for generating multimodal test instances for the 0-1 knapsack problem". Submitted to the journal of Soft Computing. Current status: Under review. Publisher: Springer.

Contents

List of Figures

List of Tables

Abstract

This research aims to investigate the multimodal properties of structural optimization using techniques from the field of evolutionary computation, specifically niching and bilevel techniques. Truss design is a well-known structural optimization problem which has important practical applications in many fields. Truss design problems are typically multimodal by nature, meaning that it offers multiple equally good design solutions with respect to the topology and/or sizes of the members, but they are evaluated to have similar or equally good objective function values. From a practical standpoint, it is desirable to find as many alternative designs as possible, rather than finding a single design, as often practiced. Niching is an intuitive way of finding multiple optimal solutions in a single optimization run. Literature shows that existing niching methods are largely designed for handling continuous optimization problems. There does not exist a well-studied niching method for constrained discrete optimization problems like truss design problems. In addition, there are no well-defined multimodal discrete benchmark problems that can be used to evaluate the reliability and robustness of such a niching method.

This thesis fills the identified research gaps by means of five major contributions. In the first contribution, we design a test suite for producing a diverse set of challenging multimodal discrete benchmark problems, which can be used for evaluating the discrete niching methods. In the second contribution, we develop a binary speciation-based PSO (B-SPSO) niching method using the concept of speciation in nature along with the binary PSO (BPSO). The results show that the proposed multimodal discrete benchmark problems are useful for the evaluation of the discrete niching methods like B-SPSO. In light of this study, a time-varying transfer function based binary PSO (TV_T-BPSO) is developed for the B-SPSO which is the third contribution of this thesis. We propose this TV_T-BPSO for maintaining a better balance between exploration/exploitation during the search process of the BPSO. The results show that the TV_T-BPSO outperforms the state-of-the-art discrete optimization methods on the large-scale 0-1 knapsack problems. The fourth contribution is to consider and formulate the truss design problem as a bilevel optimization problem. With this new formulation, truss topology can be optimized in the upper level, at the same time the size of that truss topology can be optimized in the lower level. The proposed bilevel formulation is a precursor to the development of a bilevel niching method (Bi-NM) which constitutes the fifth contribution of this thesis. The proposed

Bi-NM method performs niching at the upper level and a local search at the lower level to further refine the solutions. Extensive empirical studies are carried out to examine the accuracy, robustness, and efficiency of the proposed bilevel niching method in finding multiple topologies and their size solutions. Our results confirm that the proposed bilevel niching method is superior in all these three aspects over the state-of-the-art methods on several low to high-dimensional truss design problems.

CHAPTER 1

Introduction

1.1 Motivation

In mathematics, computer science, and operations research, optimization is the process of finding the best solution for a given problem under various circumstances [Gupta 2008, Rao and Rao 2009]. In fact, everybody does optimization to some extent. For example, we can think about choosing the right amount of crushed ice for a drink, or choosing the best route to go to work. To optimize something, the first task is to identify the objective, i.e., a quantitative measure of the performance of the problem under study. Basically, this objective depends on certain characteristics of the problem represented by variables, that may or may not be constrained. In optimization, the task is to find values for the variables that optimize the objective of the problem under consideration.

Optimization deals with the problems that are often continuous or discrete [Arora 2004]. An optimization problem with discrete variables is known as a discrete optimization problem. Discrete optimization problems concern about the efficient allocation of limited resources to fulfill the desired objective. Decision variables of such a problem are restricted to take values from bounded or discrete sets, and additional constraints limit the possible alternatives that are considered feasible. Typically, there are many possible alternatives to consider, and a goal determines which of these alternatives are the best. Continuous optimization problems are different than the discrete one in the sense that they concern about optimal setting of parameters or continuous decision variables. In this case, no limited number of alternatives exists, but optimal values for continuous variables have to be determined. In real-world, many optimization problems exist which have both continuous and discrete decision variables [Christensen 2012, Christiansen et al. 2015, Xiao et al. 2012, Ohsaki 2016, Rong et al. 2002, Parmee and Hajela 2012, Lukáš 2001, Shin et al. 2002, Zhu et al. 2016] and they are often challenging to solve [Socha 2009]. Structural optimization deals with such a challenging real-world optimization problem

which is the main concern of this thesis.

Structural optimization is a well-established research area which has interesting theoretical implications in mathematics and mechanics, as well as important practical applications in various engineering fields including the industrial design [Naceur et al. 2004], civil engineering [Huang and Xie 2010], electrical engineering [Zhou et al. 2010], automobile engineering [Fredricson et al. 2003], aerospace engineering [Krog et al. 2004], and medical engineering [Sutradhar et al. 2010]. The purpose of structural optimization is to fulfill the common need of engineers to build structures that not only satisfies their functional requirements but also perform optimally [Christensen and Klarbring 2009]. Broadly speaking, two types of structural design optimization can be found in the literature [Christensen and Klarbring 2009]: sizing optimization and topology optimization. In sizing optimization, a set of sizing parameters of the design domain may change during the optimization process without any modification of the structure topology; therefore the topology of the design domain is predefined and remains unchanged, as shown in Fig. 1.1(a). Unlike size optimization, topology optimization tries to identify an optimal design by changing the sizing parameters as well as the topology of the design domain, as shown in Fig. 1.1(b). Structural optimization provides several benefits; specifically, it reduces wastage of material, minimizes construction cost due to optimization of the amount of material requirements, minimizes construction time, provides aesthetically pleasing architecture, and obtains high-performance designs for the structures under consideration.

Several factors make structural optimization problems exceedingly difficult [Wang et al. 2004]. Firstly, structural design problems governed by the user-defined objectives and constraints [Miguel et al. 2013]. These constraints often conflict with the objective function and thus finding an optimal solution for such a problem is a challenging task. Besides, both the objective function and constraints (which are often nonlinear) cause the search space of a structural optimization problem to become multimodal, meaning that multiple optimal solutions may exist in the search space of such a problem. In this case, it is even more challenging to obtain an optimal solution of such a structural design problem, because the search process may stuck at a local optimum. Secondly, the optimal design of a structural problem not only depends on its topology optimization but also on the size optimization of that topology. This inherent relationship of structural topology and size variables strongly suggests that structural topology or sizing optimization should be performed simultaneously [Wang et al. 2004], not separately, as often practiced [Deb and Gulati 2001, Luh and Lin 2011, Li et al. 2002]. Finally, the structural sizing and topology variables may have both continuous and discrete values. Hence, optimizing these two types of variables together may cause considerable mathematical difficulties, and sometimes lead to ill-conditioning problems because their changes are of widely different orders of magnitude [Wang et al. 2004].

In the past decades, several structural optimization methods have been developed for

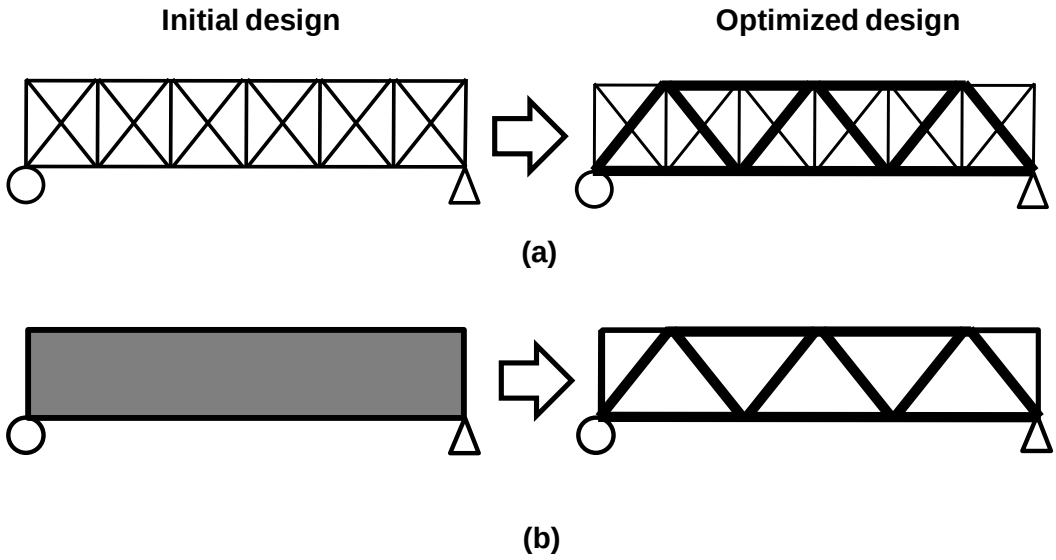


Figure 1.1: Illustration of (a) the result of sizing optimization, and (b) topology optimization of a structural design problem.

structural optimization problems [Bendsøe 1989, Xie and Steven 1993, Deb and Gulati 2001, Tai and Akhtar 2005a]. It can be observed that these existing methods are mainly designed for locating the single optimal solution. However, structural optimization problems are the non-convex optimization problems which have several linear and nonlinear constraints. Due to their non-convex property, there exist multiple equally good quality solutions in terms of the topology and size of the members, and it is desirable to locate these solutions for several reasons. Locating all the optimal/near optimal design solutions not only helps understand better the hidden relationship among found optimal solutions and facilitates sensitivity studies of the problem under investigation, but also increases the probability of finding the global optimal design solutions [Li et al. 2002]. In addition, multiple solutions can be used for designing low cost and high-performance alternative structures for the structural design problem under consideration.

In the field of evolutionary computation (EC), there has been a growing interest in applying evolutionary algorithms (EAs) to solve multimodal optimization problems. Because a multimodal optimization problem involves multiple optimal solutions, many niching methods have been developed and incorporated into evolutionary algorithms for locating such optimal solutions. A niching algorithm creates and maintains formation of several subpopulations within a single population aiming to find multiple optimal solutions in a single run [Goldberg and Richardson 1987]. Literature shows that fitness sharing is the first niching method developed by Goldberg and Richardson [Goldberg and Richardson 1987]. Since then many other niching methods have been developed in the past decades including clearing [Petrowski 1996], deterministic crowding [Mahfoud 1995], species [Li

et al. 2002], and speciation-based PSO [Li 2004]. In this thesis, these niching methods can be used as efficient tools for solving multimodal structural optimization problems. However, these niching methods have been typically designed for solving continuous problems. From a practical viewpoint, it is desirable to develop niching methods for both continuous and discrete optimization problems, such as the structural optimization problems.

One possible way of developing discrete niching methods is to extend the idea of existing continuous niching methods into discrete domains. In literature, several discrete optimization methods are available which have been developed for solving various discrete optimization problems [Pedrasa et al. 2009, Liao et al. 2007, Jarboui et al. 2008, Naeem et al. 2012, Lin et al. 2016, Han et al. 2017]. Although these methods perform well in the case of global optimization, their behaviour and performance cannot be guaranteed when applying them to multimodal discrete optimization problems. Hence, it is necessary to examine and improve these methods before using them for the development of discrete niching methods.

Artificial multimodal benchmarks are one of the primary tools that EC researchers use to demonstrate the strengths and weaknesses of a niching algorithm and to compare new niching algorithms with existing ones on a common ground. In literature, many multimodal benchmarks can be found [Goldberg 1989, Mahfoud 1995, Rönkkönen et al. 2008], which have been largely designed for the continuous optimization problems. There is no well-defined benchmark for multimodal discrete/combinatorial optimization. Although there exist many niching methods (as mentioned earlier) developed for solving the continuous optimization problems, their behaviour and performance cannot be guaranteed when applying them to discrete/combinatorial optimization problems. Therefore, it is necessary to design multimodal discrete benchmark problems to help EC researchers evaluate their niching algorithms, and further test them on structural design problems.

Problem formulation of a structural design problem is significant, as it will determine the quality of the optimization itself. In fact, optimization cannot provide useful insights of a structural design problem with an inappropriate formulation. In the past decades, several formulations have been proposed for structural optimization [Bendsøe 1989, Deb and Gulati 2001, Friedlander and Gomes 2011, Fenton et al. 2016]. It is worth to mention that all these formulations have been designed for either the structural topology optimization or size optimization. However, an optimal design of a structural design problem depends on the simultaneous topology and size optimization, because structural topology and size are two non-separable problems [Deb and Gulati 2001]. In this context, existing formulations may not be appropriate for obtaining an optimal solution of such a design problem. A more suitable formation will be required to facilitate this need of the simultaneous topology and size optimization. One of the possible solutions of this issue could be the use of the idea of bilevel optimization [Bard 1998].

Bilevel optimization is an interesting research topic in EC as well as in classic opti-

mization literature [Li et al. 2006, Zhu et al. 2006, Sinha et al. 2014, Al-Khayyal et al. 1992, Bard and Moore 1990, Aiyoshi and Shimizu 1984]. In bilevel optimization, two different levels of optimization take place, with one level (i.e., lower level) of optimization being nested within the other (i.e., upper level) [Bard 1998]. In this thesis, the idea of bilevel optimization can be used as a useful tool for solving simultaneous structural topology and size optimization. Although this idea already has been used for the structural size optimization [Kočvara 1997, Christiansen et al. 2001, Noilublao and Bureerat 2013], a significant modification is required before applying this idea effectively for the simultaneous topology and size optimization.

In the real-world, many optimization problems exist that can be considered as bilevel and multimodal by nature [Sinha et al. 2017]. Due to the multimodal property, there may exist multiple optimal solutions in the search space of a bilevel optimization problem. To find these solutions, a bilevel niching method is required. Literature shows that existing niching methods have been designed for the single level optimization problems [Pérez et al. 2003, Shir et al. 2005, Sheng et al. 2008, Chang et al. 2011]. From the practical standpoint, it is imperative to develop a bilevel niching method so as to find good quality solutions for the bilevel optimization problems such as structural design problems.

By considering the above research gaps, this research aims to develop a new structural optimization framework by incorporating the idea of the niching methods along with the idea of structural and bilevel optimization so that a diverse set of equally good quality solutions can be found for the structural design problems, with better efficiency and accuracy.

1.2 Research Objectives

The specific research objectives of this thesis are as follows:

1. To propose guidelines for designing and implementing multimodal discrete benchmark problems and to assess the robustness of a developed discrete niching method on these new benchmark problems.

2. To identify the limitations of existing discrete optimization methods, specifically, binary PSO and its popular variants in terms of providing optimal/near optimal solution of the real-world optimization problems, and to design a new binary PSO method that overcomes these limitations by means of providing a good balance between exploration and exploitation.

3. To formulate the truss design as a bilevel problem, and to design a bilevel niching method that optimizes the bilevel truss problem to find multiple truss topologies and size solutions.

4. To demonstrate the accuracy, robustness, and efficiency of the proposed bilevel niching method, compared to other well-known truss optimization methods, over various challenging low and high-dimensional truss design problems.

1.3 Methodology

Truss design is a well-known structural optimization problem where an optimal design solution depends on both of its topology and size optimization [Deb and Gulati 2001]. In truss optimization, the problem related to the topology optimization is a discrete and non-convex optimization problem which have several linear and non-linear constraints. This non-convex property causes the search space of such a problem highly multimodal. There is a good chance that multiple good quality design solutions can be found in the topology search space of truss design problems. Niching methods are well-established methods in EC which have the ability of finding multiple solutions in a single run [Goldberg 1989]. In literature, many niching methods exist [Li et al. 2017] that can be used as efficient tools for the multimodal topology optimization problems. However, these existing niching methods cannot be applicable for such a problem, because they have been designed for the continuous multimodal optimization problems. Therefore, a new niching method is urgently needed to solve the multimodal topology optimization problems.

For the proposed niching method, binary particle swarm optimization (BPSO) [Kennedy and Eberhart 1997] is selected as a potential solution to handle the truss topology optimization problems. BPSO has been shown to be effective for solving discrete optimization problems [Jordehi and Jasni 2015]. Literature shows that the niching methods with speciation concept [Li et al. 2002] can form stable niches than other existing niching methods [Li 2004]. Hence, this research will adopt the idea of speciation and BPSO to develop the proposed discrete niching method for the multimodal truss topology optimization.

Performance evaluation is an important task for the validation of any developed niching method. This can be done by conducting a comparative study between the developed niching method and state-of-the-art niching methods over well-known multimodal benchmark suites. Literature shows that existing multimodal benchmark suites [Goldberg 1989, Mahfoud 1995, Rönkkönen et al. 2008] have been largely designed for the continuous problems. There is no well-defined benchmark for multimodal discrete/combinatorial optimization that can be used for the performance evaluation of discrete niching methods. To fill this gap, this study will design a new test instance generation tool for generating multimodal discrete optimization problems. The 0-1 knapsack problem is selected as a representative combinatorial optimization problem to design this new benchmark generation tool. The 0-1 knapsack problem has been chosen because, this is a well-known NP-hard problem and widely used in practice [Mavrotas et al. 2008, Wscher et al. 2007, Higgins et al. 2008]. This research will investigate the structural relationship between

the knapsack parameters to identify the key factors that allow generating multimodal instances that facilitate this study.

As mentioned earlier, BPSO is selected as a potential tool for developing the proposed discrete niching method. However, BPSO has the limitations of providing good quality solutions for the discrete optimization problems [Ting et al. 2006, Tassopoulos and Beligiannis 2012a;b, Liu et al. 2015a, Han et al. 2017, Liu et al. 2015b, Wang et al. 2013]. Several studies have shown that the sigmoid transfer function of BPSO is one of the reasons that prevent BPSO to provide good quality solutions [Mirjalili and Lewis 2013, Bansal and Deep 2012, Liu et al. 2011; 2015b]. To overcome this issue, this study will design a new transfer function to make a good balance between exploration and exploitation of BPSO. For the design of new transfer function, we will analyse the search behaviour of existing BPSO's [Kennedy and Eberhart 1997, Mirjalili and Lewis 2013, Bansal and Deep 2012, Liu et al. 2011].

Mathematical formulation is significant in the solution process of truss design problems. In fact, an optimal solution of such a problem highly depends on its proper formulation. In literature, several formulations have been proposed for the truss optimization [Bendsøe 1989, Deb and Gulati 2001, Friedlander and Gomes 2011, Fenton et al. 2016]. However, these formulations cannot be used for the simultaneous topology and size optimization, because they have been particularly designed for the truss topology or size optimization. To address this issue, the idea of bilevel optimization [Bard 1998] will be adopted for formulating the truss design problem as a bilevel optimization problem. We have chosen bilevel optimization, because it will facilitate the simultaneous topology and size optimization of two non-separable problems like truss topology and size problems.

Like truss design problems, many other optimization problems can be viewed as a bilevel optimization problem [Christiansen et al. 2015, Xiao et al. 2012, Ohsaki 2016, Rong et al. 2002, Parmee and Hajela 2012, Lukáš 2001, Shin et al. 2002]. As most of these problems are non-convex optimization problems, their search spaces may contain more than one equally good quality solution. Niching methods are required for finding multiple good quality solutions of these problems. Literature shows that existing niching methods have been developed for the problems which have only one level of optimization task [Pérez et al. 2003, Shir et al. 2005, Sheng et al. 2008, Chang et al. 2011]. To fill this gap, this study will develop a bilevel niching method, specifically for the bilevel truss design problems. For the proposed bilevel niching method, we will adopt the proposed discrete niching method and a standard PSO [Kennedy and Eberhart 1997]. The proposed bilevel niching method will be used to find multiple topology and size solutions of truss design problems. Finally, we will evaluate the proposed bilevel niching method based on the following performance criteria: the number of design solutions, the final quality of the solutions, and efficiency, and compared that with several state-of-the-art structural optimization methods [Deb and Gulati 2001, Luh and Lin 2011, Li 2015].

1.4 Contributions

This study makes the following original research contributions in structural design optimization as well as evolutionary algorithms.

- Design and development of a benchmark instance generation tool for generating a diverse set of multimodal 0-1 knapsack problems. This will facilitate evaluation of discrete niching methods.
- Propose a new discrete niching method that can find multiple design solutions for the truss topology optimization problems.
- Propose a new binary PSO, namely the time-varying transfer function based BPSO (TV_T-BPSO) for obtaining the better quality solution of discrete problems by means of balancing the exploration and exploitation ability of BPSO.
- Formulation of the truss design problem as a bilevel optimization problem to facilitate the simultaneous topology and size optimization.
- Development of a bilevel niching method that can identify the multiple topology and size solutions simultaneously for the bilevel truss design problems.

1.5 Outline of the Thesis

The remaining of this thesis is organized as follows. The background materials of this thesis are provided in Chapter 2. Chapter 3 contains a benchmark suite for multimodal discrete optimization problems. This chapter also contains a binary niching method namely B-SPSO which is developed based on the concept of speciation and the original binary PSO (BPSO). In addition, this chapter shows how the newly proposed benchmarks pose a major challenge to the proposed B-SPSO method. In Chapter 4, a time-varying transfer function based BPSO (TV_T-BPSO) is proposed in light of the lessons that we learn from Chapter 3. In Chapter 5, we first formulate the truss problem as a bilevel problem. Based on this formulation, a bilevel niching method is proposed which is developed based on the improved B-SPSO and a standard PSO. Numerous examples are provided in this chapter to show the efficiency and accuracy of the proposed bilevel niching method over low- and high-dimensional challenging truss design problems. Finally, we conclude this thesis with a summary of major findings in Chapter 6.

CHAPTER 2

Background

2.1 Introduction

This chapter contains the background materials that are used throughout this thesis. Particularly, the idea of general optimization which can be applied to problems with different types of variables is discussed. The popular metaheuristic algorithms that are related to this study are introduced. An overview of structural optimization and its key optimization techniques are provided. The basic ideas of popular multimodal optimization techniques are introduced. An overview of existing continuous multimodal benchmark functions and the needs of the discrete multimodal benchmark functions are also discussed. Finally, bilevel optimization and its application to structural optimization are discussed.

2.2 Optimization

Optimization is the process of finding optimal solutions for a given problem under certain defined circumstances [Rao and Rao 2009]. In fact, we all do it in our daily life to some extent. Be it thinking about the choice of the best route between two points or arranging furniture in such a way that the space in a room can be utilized properly. In the case of a manufacturing system, we can describe the optimization in the following way. During the development of such a system, it is necessary to make the different technological and managerial decisions at several stages. The purpose of making all these decisions is to minimize the effort required or to maximize the desired profit. Since the effort required and/or the benefit desired in any practical situation can be expressed as a function of certain decision variables, optimization can be defined as the process of finding the conditions that give the maximum (or minimum) value (x^*) of a function, as shown in Fig. 2.1.

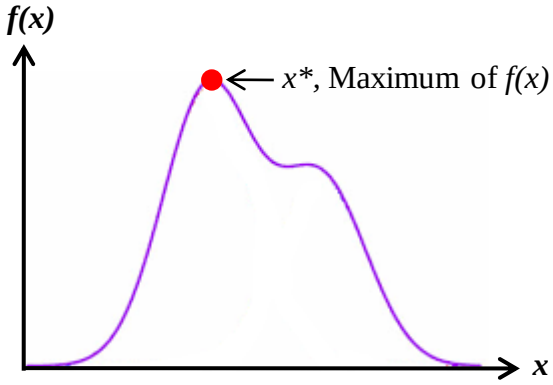


Figure 2.1: Illustration of the maximum (x^*) of an optimization function $f(x)$ that tries to find by optimization.

In mathematics and computer science, an optimization problem is a problem of finding the best solution among all of its feasible solutions. To optimize a problem, it is necessary to be familiar with three key concepts related to optimization i.e., *decision vector*, *constraints*, and *objective function*, which are defined in the following:

- **Decision Vector:** In practice, optimization problems are defined by a set of parameters in which all of them change their values during the optimization process. Thus, all these parameters are treated as variables of the optimization problem and are called decision variables x_i, i=1, 2, ..., n. The variables of an optimization problem are collectively termed as a decision vector $\mathbf{x}= \{x_1, x_2, \ldots, x_n\}$.

- **Constraints:** In many practical problems, the values of the decision variables cannot be chosen arbitrarily; rather, they have to satisfy certain specified functional and other requirements. The restrictions that must be satisfied to produce an acceptable result are collectively called the constraints of the optimization problem.

- **Objective Function:** The goal of a typical optimization method is to find an acceptable solution that must fulfill the requirements of the problem. In general, the search space of a problem contains more than one acceptable solutions, and the optimization method aims to find the best one among these feasible solutions. Certain criteria have to be imposed for comparing the different solutions or for selecting the best solution. The criterion with respect to which the problem is optimized, when expressed as a function of the decision variables, is known as the objective function. The choice of objective function is determined by the problem properties and the model designer.

Considering the above, an optimization problem can be defined as follows.

$$\begin{aligned} \text{Find} \quad & \mathbf{x} = \begin{Bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_n \end{Bmatrix} \text{ which maximizes } f(\mathbf{x}) \\ \text{subject to} \quad & g_i(\mathbf{x}) \leq 0,\ i = 1, \ldots, m \\ & h_j(\mathbf{x}) = 0,\ j = 1, \ldots, q, \end{aligned} \tag{2.1}$$

where $\mathbf{x}$ is an n-dimensional decision vector, $f(\mathbf{x})$ is the objective function, and $g_i(\mathbf{x})$ and $h_j(\mathbf{x})$ are inequality and equality constraints, respectively. The problem presented in Eq. (2.1) is a constrained optimization problem.

2.2.1 Classification of Optimization Problems

Optimization problems can be classified into different categories depending on whether they are constrained or unconstrained, the variables are separable or non-separable, the variables are continuous or discrete, and the variables are mixed i.e., continuous and discrete. A brief description of these categories is provided below:

- *Continuous and discrete problems*: An optimization problem with continuous variables is known as a continuous optimization problem. For example, we can consider the problem stated in Eq. (2.1) as a continuous optimization problem when its decision variable $\mathbf{x}$ is treated as a continuous variable. In contrast, an optimization problem with discrete variables is known as the discrete optimization problem. The variables of a discrete optimization problem can take a discrete value from a set. For example, if we restrict the variables such that they can only take on values from the set $\{0,\ 1,\ 5,\ 6\}$ then the problem stated in Eq. (2.1) can be said a discrete optimization problem.

- *Mixed variable problems*: Many real-world optimization problems can be modeled using combinations of continuous and discrete variables which are generally called mixed-variable optimization problems. A mixed variable problem can be defined in the following way. Let $R = (\mathbf{a}, \mathbf{g}, f)$ be a mixed-variable model where $\mathbf{a}$ represents a finite set of both discrete and continuous decision variables, $\mathbf{g}$ represents the set of constraints among these variables, and f denotes an objective function to be minimized or maximized. The search space $\mathbf{a}$ consists of $n = d + r$ variables x_i $(i = 1, 2, \ldots, n)$ of which d is the number of discrete variables and r is the number of continuous variables. A feasible solution $a \in \mathbf{a}$ is a solution that satisfies all constraints in the set $\mathbf{g}$. A global optimum $a^* \in \mathbf{a}$ is a feasible solution that satisfies $f(a^*) \leq f(\mathbf{a})\ \forall a \in \mathbf{a}$.

- *Constrained and unconstrained problems*: As mentioned earlier, an optimization problem can be classified as a constrained or unconstrained optimization problem by determining that whether the constraints exist in the problem or not.

- *Separable and non-separable problems*: A function $f(\mathbf{x})$ is said to fully separable if it can be expressed as the sum of n single-variable functions, $f_1(x_1)$, $f_2(x_2)$, . . . , $f_n(x_n)$, that is,

$$f(\mathbf{x}) = \sum_{i=1}^{n} f_i(x_i). \tag{2.2}$$

In a fully separable problem, the objective function and the constraints are totally separable from each other to each other and it can be expressed as

$$\begin{aligned} \text{Find } \mathbf{x} \text{ which maximizes} \quad & f(\mathbf{x}) = \sum_{i=1}^{n} f_i(x_i) \\ \text{subject to} \quad & g_j(\mathbf{x}) = \sum_{i=1}^{n} g_{ij}(x_i) \leq b_j,\ j = 1, 2, \ldots, m \end{aligned} \tag{2.3}$$

where b_j is a constant. A function f said to be partially separable if it is the sum of element functions,

$$f(\mathbf{x}) = \sum_{i=1}^{m} f_i(x), \tag{2.4}$$

where each individual function f_i is a function depending on some of the components x_j, $j \in I_i$ for some index sets $I_i \subseteq \{1, ..., n\}(i = 1, ..., m)$. Finally, a problem is called a non-separable problem, if its variables show inter-relation among themselves or are not independent.

2.3 Metaheuristic Algorithms

Optimization algorithms are generally divided into two categories. These are deterministic algorithms and stochastic algorithms. Deterministic algorithms follow a precise sequence of actions. Thus, design variables and objective function can have the same values and repeat the same solution route. On the other hand, a stochastic algorithms always involves randomness. Stochastic algorithms are popular for the optimization problems that are described in the previous section. Stochastic algorithms can be classified into two categories such as heuristic and metaheuristic algorithms. Heuristic algorithms are problem dependent algorithms where each optimization problem has its own heuristic methods and these heuristics cannot be applied to other types of optimization problems. On the other hand, metaheuristic based algorithms consist of a general solver template which can be used for different types of optimization problems by properly tuning its operators and configuring its parameters. In literature, two different types of metaheuristic algorithms exist namely the Evolutionary Algorithms (EAs) and Swarm Intelligence Algorithms.

Evolutionary algorithms optimize a problem based on the evolutionary principle of survival of the fittest. Particularly, an EA begins with a set of randomly generated population individuals (solutions). At each generation, the EA modifies the key characteristics of the current population in order to form a new population that will be selected based on the natural selection principle. Examples of evolutionary algorithms include genetic algorithm (GA) [Holland 1992b], and genetic programming (GP) [Koza 1992]. *Swarm intelligence algorithms* are inspired by the collective behavior of a group of animals or insects when searching for food. At each iteration, the swarm intelligence algorithm constructs the solutions based on the historical information attained from previous generations. Examples of swarm intelligence algorithms include the particle swarm optimization (PSO) [Kennedy and Eberhart 1995], Ant colony optimization (ACO) [Dorigo et al. 1996], and artificial bee colony (ABC) [Karaboga 2005]. In the followings, we will provide detailed descriptions of these optimization algorithms.

2.3.1 Genetic Algorithm (GA)

Genetic algorithm works based on the principles of Darwin's evolution theory and natural selection of biological systems [Holland 1992b]. Basically, a genetic algorithm starts optimization process with a set of randomly generated a population of individuals. Each of these population individuals is a chromosome encoding a different candidate solution for the problem under consideration. During each generation, chromosomes are evaluated using a fitness function. To create the next generation, new chromosomes (known as offspring) are created by the crossover operation. After that, the mutation operator randomly mutates on a gene of a chromosome. Note that mutation is a crucial step in GA since it ensures that the population remains diverse and always contains new genes for future generations. Finally, a new generation is formed by selecting the fittest chromosomes (from both the parents and offspring) and rejecting others so as to keep the population size constant. After several generations, the GA converges to the best chromosome which represents the optimal or near optimal solution of the problem. The working steps of GA can be described as follows:

- Step 1: Generate random population of n chromosomes.
- Step 2: Evaluate the fitness $f(x)$ of each chromosome x in the population.
- Step 3: Create a new population by repeating the followings:
  - Step 3.1: Select two fittest parent chromosomes from the population of the current generation.
  - Step 3.2: Perform crossover with probability p_c to selected fitter parent individuals to form new offspring (children).

  - Step 3.3: Perform mutation with probability p_m to offspring.
  - Step 3.4: Apply the selection operation to parents and offspring to form the new population for the next generation.

- Step 4: Stop the algorithm when the termination criteria is met, otherwise go to Step 2.

The genetic algorithm has been successfully applied to problems in a variety of fields of study, and their popularity continues to increase due to their effectiveness, their applicability, and their ease of use. The well-known applications of GA including scheduling [Murata et al. 1996, Gonçalves et al. 2005, Asadzadeh 2015], vehicle routing [Zuo et al. 2015, Mohammed et al. 2017, Xiao and Konak 2017], reliability design [Taboada et al. 2008, Ye et al. 2010, Sahoo et al. 2012, Duan et al. 2015], transportation [Vignaux and Michalewicz 1991, Gen et al. 2006, Jo et al. 2007], engineering design [Gen and Cheng 2000, Deb and Gulati 2001, Li 2015], and many others.

2.3.2 Particle Swarm Optimization (PSO)

In [Kennedy and Eberhart 1995], the particle swarm optimization (PSO) is proposed to optimize the continuous optimization problems. In PSO, the particles (individuals of a swarm) are first initialized by placing them randomly in the d-dimensional search space and the search for an optimal solution is done by updating individual over successive iterations. At each iteration step, the i-th particle knows two pieces of information: its personal best position $\mathbf{p}_i$=$(p_{i1}, \ldots, p_{id})$, which is the best position it has visited so far (its memory), and the global best position of the whole swarm $\mathbf{g}$=$(g_1, \ldots, g_d)$. At each iteration, the velocity $\mathbf{v}_i$=$(v_{i1}, \ldots v_{id})$ of i-th particle of the swarm is modified by the following equation:

$$\mathbf{v}_i^{k+1} = \mathbf{v}_i^k + c_1 r_1^k (\mathbf{p}_i^k - \mathbf{x}_i^k) + c_2 r_2^k (\mathbf{g}^k - \mathbf{x}_i^k), \tag{2.5}$$

where k represents the current iteration, r_1 and r_2 are two random numbers drawn from the uniformly distribution $\mathcal{U}(0,1)$, and $c_1 > 0$ and $c_2 > 0$ are the cognitive and social coefficients, respectively.

In practice, PSO often uses the problem-dependent velocity clamping techniques (e.g. [Shahzad et al. 2009]) to prevent too large velocities. More specifically, the velocity $\mathbf{v}_i^{k+1}$ is bounded by a threshold v_{max} as follows:

$$\mathbf{v}_i^{k+1} = \begin{cases} v_{max}, & \text{if } \mathbf{v}_i^{k+1} > v_{max}, \\ -v_{max}, & \text{if } \mathbf{v}_i^{k+1} < v_{max}. \end{cases} \tag{2.6}$$

Once the velocity has been clamped, the position $\mathbf{x}_i$ of i-th particle is updated according to the following equation:

$$\mathbf{x}_i^{k+1} = \mathbf{x}_i^k + \mathbf{v}_i^{k+1} \tag{2.7}$$

Algorithm 1 Pseudocode for the particle swarm optimization.

Require: v_{max} and x_{max} ▷ the upper bound of velocity and position
Require: $iter_{max}$ ▷ the maximum number of iterations
Require: $popSize$ ▷ the population size

$iter = 0$;
//randomly initialize i-th particle velocity, position, and personal best
for i=1 to $popSize$ **do**
    $\mathbf{x}_i \in \{\text{-}v_{max}, v_{max}\}$;
    $\mathbf{x}_i \in \{\text{-}x_{max}, x_{max}\}$;
    $\mathbf{p}_i \leftarrow \mathbf{x}_i$;
end for
//main loop
repeat
    for i=1 to $popSize$ **do**
        evaluate fitness $f(\mathbf{x}_i)$;
        if $f(\mathbf{x}_i) > f(\mathbf{p}_i)$ **then**
            $\mathbf{p}_i \leftarrow \mathbf{x}_i$;
        end if
        if $f(\mathbf{x}_i) > f(\mathbf{g})$ **then**
            $\mathbf{g} \leftarrow \mathbf{x}_i$;
            $f(\mathbf{g}) \leftarrow f(\mathbf{x}_i)$;
        end if
        update $\mathbf{v}_i$ using Eq. (2.5);
        update $\mathbf{x}_i$ using Eq. (2.7);
    end for
    $iter = iter$+1;
until $iter < iter_{max}$;

We provide the pseudo-code for the particle swarm optimization in Algorithm 1. In literature, two common approaches can be found for choosing the global best position, known as *gbest* and *lbest* methods. In the *gbest* approach, the position of each particle in the search space is influenced by the best-fit particle of the entire swarm; whereas the *lbest* approach only allows each particle to be influenced by a fitter particle chosen from its local neighbourhood [Kennedy 2011].

Like GA, PSO has been successfully applied to problems in a variety of fields of study. A comprehensive survey PSO to the application of various problems can be found in [Zhang et al. 2015, Bonyadi and Michalewicz 2017].

2.3.3 Tabu Search (TS)

Tabu search works by tracking the regions of the solution space that have already been visited. Tabu search tracks the visited regions in order to avoid repeating the search in the same areas again [Glover 1989]. Tabu search starts with a random initial solution and successively moves to one of the neighbors of the current solution. In tabu search, a tabu list (a special short-term memory) is used to store the previously visited solutions that are marked as prohibited moves. During the local search, this method examines only those solutions that are not in tabu list and satisfy at least one predefined aspiration criteria.

A common aspiration criterion of tabu search is better fitness. The basic working steps of tabu search algorithm are described in the following:

- Step 1: Generate an initial solution x.
- Step 2: Set the tabu list $T := \emptyset$;
- Step 3 While the set of candidate solutions $\mathbf{X}^*$ is not complete.
  - Step 3.1: Generate candidate solution x^* from current solution x.
  - Step 3.2: if x^* is not in tabu list and at least one aspiration criterion is satisfied, then add x^* to $\mathbf{X}^*$.
- Step 4: Select the best candidate solution x^* in X.
- Step 5: If $f(x^*)$ is better than $f(x)$, then $x := x^*$.
- Step 6: Update the tabu list and aspiration criteria.
- Step 7: Stop when the termination criteria are met, otherwise go to Step 3.

Tabu search algorithm is another popular algorithm which has been applied to various optimization problems. The well-known applications of Tabu search algorithm are scheduling and routing Problem [Brandimarte 1993, Gendreau et al. 1994b], assignment problem [Skorin-Kapov 1990, Misevicius 2005, Acan and Ünveren 2015], and facility layout problem [Chiang and Kouvelis 1996, Samarghandi and Eshghi 2010, Bozorgi et al. 2015].

2.3.4 Binary Particle Swarm Optimization (BPSO)

Kennedy and Eberhart [Kennedy and Eberhart 1997] developed the first binary PSO to tackle the discrete/combinatorial optimization problems. The BPSO primarily extended the basic concept of the original PSO by using the sigmoid transfer function to transform the value of velocity from the continuous space into binary space. In this BPSO, at each iteration, the velocity $\mathbf{v}_i$ of i-th particle is modified according to Eq. (2.5). After updating the velocity, the position $\mathbf{x}_i$ i-th particle is updated according to the following:

$$\mathbf{x}_i^{k+1} = \begin{cases} 0 & \text{if } rand() \geq S_T(\mathbf{v}_i^{k+1}), \\ 1 & \text{otherwise,} \end{cases} \tag{2.8}$$

where k represents the current iteration, $S_T(\mathbf{v}_i^{k+1})$ is a sigmoid transfer function which denotes the probability (in the range of [0,1]) for a bit in a binary string that takes the value 0 or 1:

$$S_T(\mathbf{v}_i^{k+1}) = \frac{1}{1 + e^{-\mathbf{v}_i^{k+1}}}. \tag{2.9}$$

Below we provide the basic working steps for the binary particle swarm optimization.

- Step 1: Randomly initialize the position and velocity of the particles of a swarm. Repeat the following steps until the termination criteria met.
- Step 2: Evaluate the fitness of all particles.
- Step 3: Calculate the personal best position for all particles.
- Step 4: Calculate the global best position of the whole swarm.
- Step 5: Update position and velocity of all particles according to Eq. (2.5) and Eq. (2.8), respectively.

Binary particle swarm optimization (BPSO) is one popular metaheuristic algorithm which has been used for solving various types of discrete optimization problems [Pedrasa et al. 2009, Liao et al. 2007, Jarboui et al. 2008, Naeem et al. 2012, Lin et al. 2016, Han et al. 2017]. However, it has been observed that BPSO seems to be lacking of the exploration capability that is needed for obtaining high-quality solutions [Bansal and Deep 2012, Mirjalili and Lewis 2013, Liu et al. 2011]. Many BPSO variants have been developed to tackle this issue [Bansal and Deep 2012, Mirjalili and Lewis 2013, Liu et al. 2011, Ting et al. 2006, Tassopoulos and Beligiannis 2012a;b, Liu et al. 2015a;b, Wang et al. 2013]. However, the existing BPSO variants still have the issue of maintaining a good balance between exploration and exploitation ability of BPSO's.

2.3.5 Ant Colony Optimization

The ant colony optimization algorithm is inspired by the observation of real ant colonies [Dorigo et al. 2004]. Ants are social insects, meaning that they like to live in colonies and their food foraging behaviours are driven by the interactions among ants as well as interactions between ants and the environment. An important and interesting behavior of ant colonies is their foraging behavior, and, in particular, how ants can find shortest paths between food sources and their nest (see Fig. 2.2). While walking from food sources to the nest and vice versa, ants deposit on the path a substance called pheromone, forming in this way a pheromone trail. Ants can smell pheromone and, when choosing their way, they tend to follow the paths marked by stronger pheromone concentrations. The pheromone trail allows the ants to find their way back to the food source (or to the nest). Also, it can be used by other ants to find the location of the food sources found by their nestmates.

Informally, an ant colony algorithm consists of three procedures: ConstructAntSolutions, UpdatePhenomenons, and DaemonsActions [Dorigo et al. 1999]. Each of these steps is described below:

- *ConstructAntSolutions*: This procedure manages a colony of ants that concurrently and synchronously visit adjacent states of the considered problem by moving through

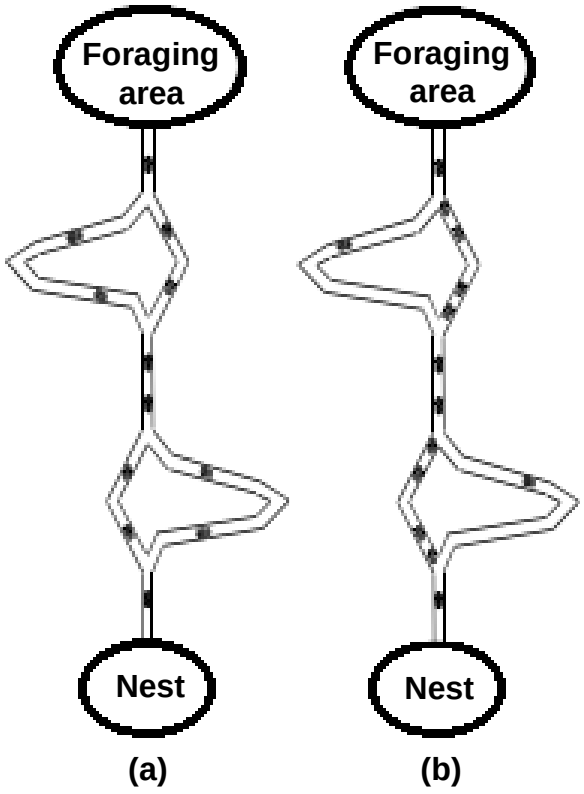


Figure 2.2: Illustrations of (a) ants start exploring the double paths, and (b) most of the ants choose the shortest path [Dorigo et al. 1999].

neighbour nodes of the problem construction graph. They move by applying stochastic local decision policy that makes use of pheromone trials and heuristic information. In this way, ants incrementally build solutions to the problem. Once an ant has built a solution, the ant evaluates the partial solution that will be used by the UpdatePheromones procedure to decide how much pheromone to deposit.

- *UpdatePheromones*: UpdatePheromones is the process by which the pheromone trial are modified. The trail value can either increase, as ants deposit pheromone on the components or connections they use, or decrease, due to pheromone evaporation. From the practical viewpoint, the deposit of new pheromone increases the probability that those components/connections that were either used by many ants or that were used by at least one ant which produced a very good solution. In addition, pheromone evaporation implements a useful form of forgetting; it avoids a too rapid convergence of the algorithm toward a suboptimal region, therefore favoring the exploration of new areas of the search space.

- *DeamonActions*: This procedure is used to implement centralized actions which

cannot be performed by single ants. An example of daemon action is the activation of a local search procedure that can be used to decide whether it is useful or not to deposit additional pheromone to bias the search process from a non-local perspective. As a practical example, the daemon can observe the path found by each ant in the colony and select one or few ants which are then allowed to deposit additional pheromone or the components/connections they used.

ACO is one of the popular optimization algorithms which has been widely used for various problems. The well-known applications of ACO are travelling salesman problem [Dorigo and Gambardella 1997, Mavrovouniotis and Yang 2013, Escario et al. 2015], scheduling problem [Chen and Zhang 2009, Merkle et al. 2002], clustering [Shelokar et al. 2004, Runkler 2005], and routing [Bell and McMullen 2004].

Basically, the above metaheuristic algorithms have been designed to tackle complex optimization problems where other optimization methods have failed to be effective. It is worth to mention that the performance of continuous metaheuristic algorithms have been significantly improved over the past decades. However, the performance of discrete metaheuristic algorithms cannot reach the level that has been achieved by the continuous metaheuristic algorithms. Because, most of these methods cannot maintaining a better balance between exploration and exploitation which is the key to find a good quality solution.

2.4 Structural Optimization

A structure in mechanics is defined as "any assemblage of materials which is intended to sustain loads" [Gordon 2012]. On the other hand, optimization can be defined as the process of finding the thing which is the best in a different manner. Thus, structural optimization is the process of finding an assemblage of materials sustains loads some measurable and optimal ways. For clarity, consider the Fig. 2.3 where a load F needs to be transmitted from an area in space to a rigid support. With this, we want to find a structure that performs this task in the best way. The term "best" could be interpreted as to make the structure as light as possible. Another interpretation of the "best" could be to make the structure as stiff as possible or it becomes insensitive to buckling as possible. It is clear that such minimization or maximization cannot be achieved without imposing any constraints. For example, if there is no limitation (i.e., constraint) imposed on the amount of the materials that need to be used, the structure can be made stiff without limit, but the optimization problem would be a problem without a well-defined feasible solution.

In structural optimization, the weight, stress, displacements and/or the geometry are considered as the constraints. For the optimization purpose, these constraints can be used as the measure of "best" that is, as objective functions. Thus, one can formulate the

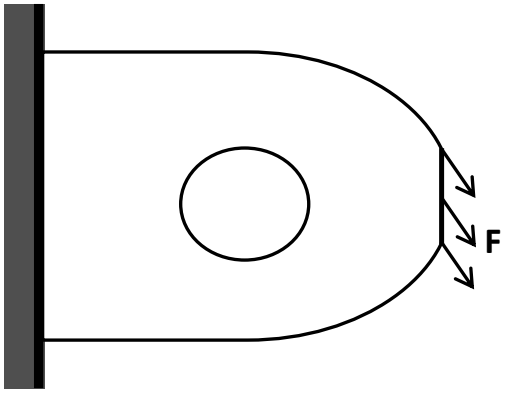


Figure 2.3: Structural optimization problem. Find the structure which best transmits the load F to the support.

structural optimization problem by using one of these constraints as an objective function that needs to be minimized/maximized and rest of them can be used as the constraints.

To perform optimization, the first task is to define the design domain of the structures that are going to be optimized. After that, the structural design problem is formulated and an optimization technique is applied to solve the problem. The general structural optimization problem can be defined as [Christensen and Klarbring 2009]:

$$(\mathbb{SO}) \quad \begin{cases} \text{minimize} & f(\mathbf{x},\mathbf{y}) \text{ with respect to } \mathbf{x} \text{ and } \mathbf{y} \\ \text{subject to} & \begin{cases} \text{behavioral constraints on } \mathbf{y}, \\ \text{design constraints on } \mathbf{x} \\ \text{equilibrium constraint}, \end{cases} \end{cases} \tag{2.10}$$

where f defines the objective function which measures the quality of a solution, $\mathbf{x}$ represents the design variable which can be a vector that describe the design, and $\mathbf{y}$ represents a state variable. For a given design $\mathbf{x}$, $\mathbf{y}$ represents the response of the structure. For a mechanical structure, response means displacement, stress, strain or force.

For structural optimization, two approaches are used commonly for defining the design domain. The first one is the *material distribution approach* [Xie and Steven 1993] and the second one is the *ground structure approach* [Deb and Gulati 2001]. In material distribution approach, the design domain is discretized into a fixed number of connected blocks, as shown in Fig. 2.4(a). With this arrangement, the optimal design of a structure is determined by removing the unnecessary materials from the design domain, according to the applied boundary conditions. In the ground structure approach, every node in the design domain is connected to all other nodes by structural elements, such as the frame, beam, and/or truss elements, as shown in Fig. 2.4(b). Note that a ground structure is a complete truss with all possible member connections among all nodes (basic or non-basic)

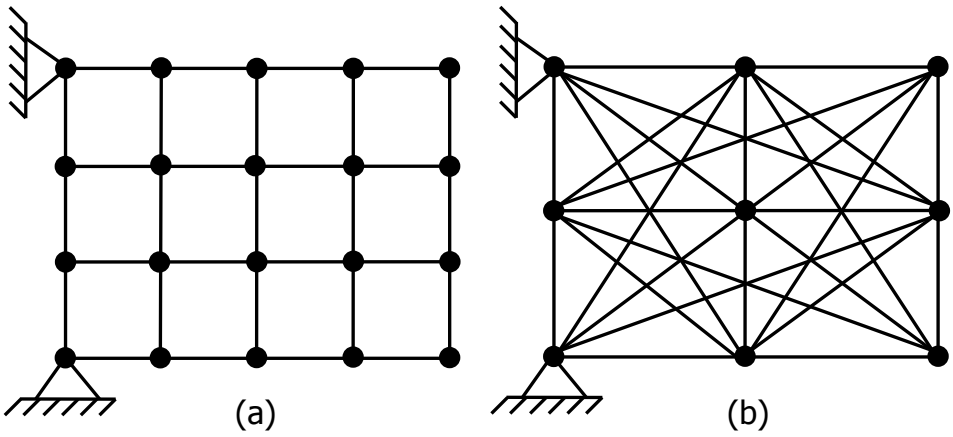


Figure 2.4: Design domain representations (a) material distribution and (b) ground structure methods.

in the structure [Deb and Gulati 2001]. Thus, in a truss having n nodes, there are a total of m=(^{n}2) different members possible. A ground structure is a collection of all these members. By the ground structure approach, the optimal design is determined by finding a subset of structural members and their corresponding cross-sectional areas. There are several advantages and disadvantages for each approach. In general, the ground structure approach requires a smaller model with lower computational efforts than the material distribution method in obtaining a meaningful solution. Therefore, the ground-structure approach is considered here in this thesis in order to achieve our research objectives.

2.4.1 Topology, Size, and Shape optimization

In literature, mainly three different types of structural optimization (based on the ground structure model) can be found [Christensen and Klarbring 2009]:

- *Topology optimization*: This is the most basic form of structural optimization. In this case, the members of a ground structure are treated as the design variables. With this, the topology optimization aims to select a subset of the members of a ground structure to form a stable truss topology that satisfies some user-defined constraints. For the illustration purpose, a topology optimization problem for a truss structure is shown in Fig. 2.5.

- *Sizing optimization*: The sizing optimization finds the member's cross-sectional areas of bars in a given ground structure in an optimal way. Sizing optimization for a truss structure is illustrated in Fig. 2.6.

- *Shape optimization*: Shape optimization is different from the previous two optimizations. Shape optimization aims to optimize the nodal coordinates of the selected

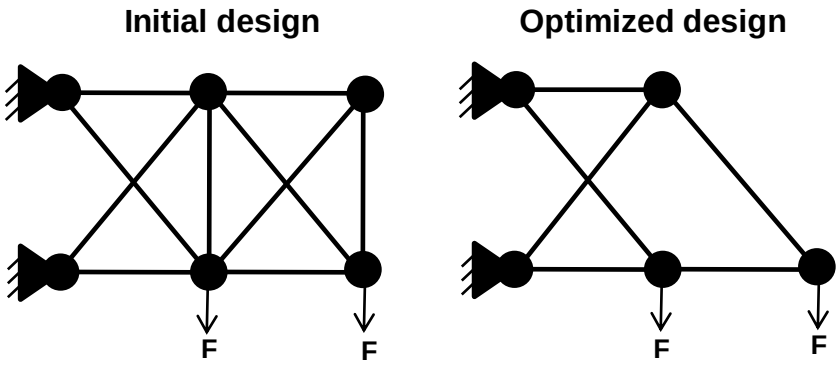


Figure 2.5: Structural topology optimization is carried out by optimizing the members of a ground structure. In this case, the cross-sectional area of all the truss members are assumed to be equal and fixed during the optimization.

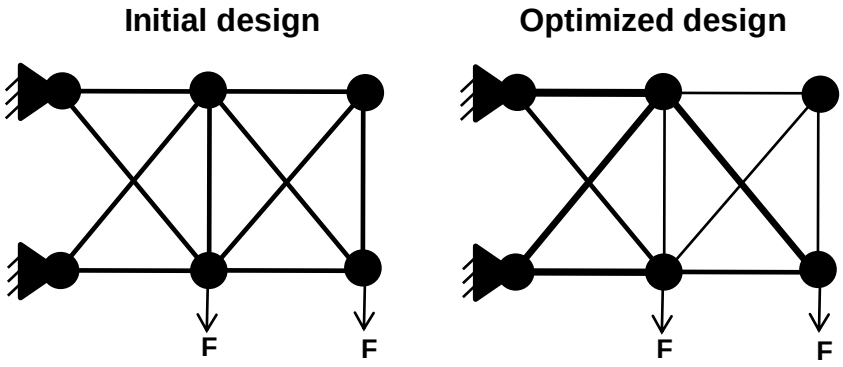


Figure 2.6: Sizing optimization is carried out by optimizing the member cross-sectional areas of bars in a ground structure.

nodes of a ground structure. As a result, the length of the members that are connected with the selected nodes would be changed in optimized design, as illustrated in Fig. 2.7.

2.4.2 Simultaneous Topology and Size Optimization

Structural optimization problems governed by the user-defined objectives and constraints [Miguel et al. 2013]. These constraints often conflict with the objective function and thus finding an optimal solution for such a problem is a challenging task. Besides, both the objective function and constraints (which are often nonlinear) cause the search space of a structural optimization problem to be multimodal. In this scenario, it is even more challenging to find an optimal solution of such a problem, because the search process may easily get stuck in a locally optimal solution. Another challenge is that the optimal

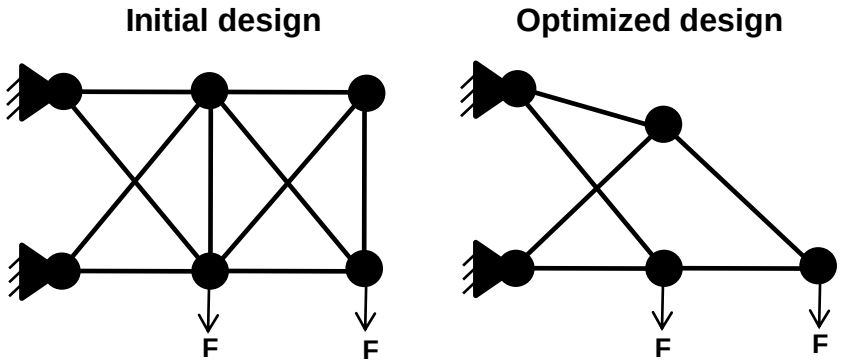


Figure 2.7: Shape optimization is carried out by optimizing the nodal coordinates of a ground structure. In this case, the cross-sectional area of all the truss members are assumed to be equal and fixed during the optimization run.

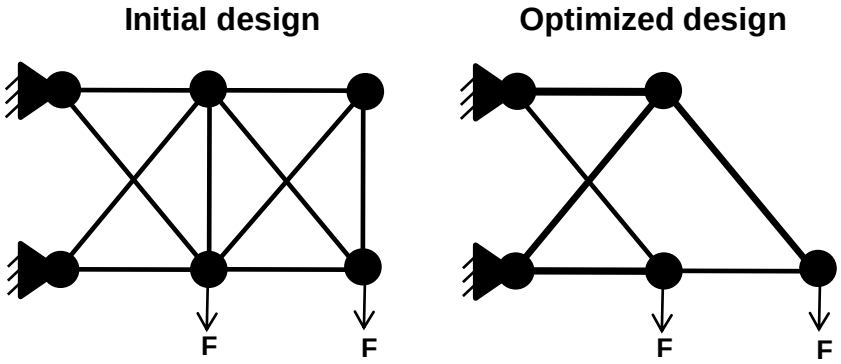


Figure 2.8: Illustration of the simultaneous topology and size optimization of a ground structure.

design of a structural optimization problem not only depends on its topology optimization, but also depends on the size optimization [Wang et al. 2004]. Therefore, it is necessary to perform the structural topology and size optimization simultaneously in order to find an optimal/near optimal design solution. An example of simultaneous topology and size optimization of a ground structure is illustrated in Fig. 2.8.

Proper formulation is one of the key steps for the simultaneous topology and size optimization of a structural design problem, as it determines the quality of the optimization itself. In literature, various formulations have been proposed for the structural optimization. For example, Bendsøe and Martin [Bendsøe 1989] performs the structural optimization using an elasticity tensor formulation. Deb and Gulati [Deb and Gulati 2001] performs the structural optimization using a nonlinear programming formulation. A bi-objective formulation is introduced for the structural optimization in [Noilublao and

Bureerat 2011; 2013]. Another formulation based on the force method is proposed in [Rahami et al. 2008]. A bilevel formulation for the structural design problems is presented in [Friedlander and Gomes 2011]. Recently, a new formulation so-called grammatical evolution is proposed in [Fenton et al. 2016] for the structural optimization problems. Among these formulations, the following non-linear programming formulation is frequently used for the structural optimization [Deb and Gulati 2001, Luh and Lin 2011; 2008, Dominguez et al. 2006]:

$$\begin{aligned} \text{Minimize } & W(\mathbf{A}) = \sum_{j=1}^{n_m} \rho_j \ell_j A_j \\ \text{Subject to } & G_1 : \textit{Truss contains all the basic nodes,} \\ & G_2 : \textit{Truss is kinematically stable,} \\ & G_3 : S_j \geq \sigma_j(\mathbf{A}, \xi),\ j = 1, 2, .., n_m, \\ & G_4 : \delta_i^{max} \geq \delta_i(\mathbf{A}, \xi),\ i = 1, 2, .., n_n, \\ & G_5 : A_{min} \leq A_j \leq A_{max},\ j = 1, 2, .., n_m, \\ & G_6 : \xi_{min} \leq \xi_i \leq \xi_{max},\ i = 1, 2, .., n_n, \end{aligned} \tag{2.11}$$

where W denotes the weights of the truss, $\mathbf{A}$ and ξ represent the set of cross-sectional areas and the set of nodal coordinates, respectively, n_m and n_n represent the number of members and nodes of a given ground structure, respectively. The parameters ρ_j and ℓ_j are the material density and length of the j-th member, respectively. The parameter A_j represents the cross-sectional area of the j-th member. Constraint G_1 ensures that the truss contains all the basic nodes. Constraint G_2 ensures the kinematical stability of the truss which can be checked by determining the positive definiteness of the stiffness matrix. Constraint G_3 ensures that the stress σ_j of j-th member is less than or equal to the allowable stress S_j. Constraint G_4 ensures that the displacement δ_i of i-th node is less than or equal to the allowable displacement δ_i^{max}. Constraint G_5 ensures that the value of A_j is within the limits $[A_{min}, A_{max}]$. Finally, constraint G_6 ensures that the coordinates of the i-th node ξ_i is within the limits $[\xi_{min},\xi_{max}]$.

Note that the existing formulations including Eq. (2.11) have been developed based on the assumption that the structural topology and size are two separable problems. In reality, structural topology and size problems are two non-separable problems [Deb and Gulati 2001], thus the use of these existing formulations may not be appropriate for the simultaneous topology and size optimization.

2.5 Structural Optimization Methods

In this section, a brief survey of popular numerical and metaheuristic-based structural optimization methods is provided.

2.5.1 Numerical Methods

Several numerical structural optimization methods have been proposed in literature; these methods include the Solid Isotropic Material with Penalization (SIMP), Evolutionary Structural Optimization (ESO), and Level-set method.

Solid Isotropic Material with Penalization (SIMP)

Solid Isotropic Microstructure with Penalization (SIMP) method is first introduced in [Bendsøe 1989]. The SIMP method operates on a fixed domain of finite elements with the aim of minimizing an objective function by identifying whether each element should consist of solid material or void. This method assumes that the material property of each finite element is constant, whereas the density of finite elements is defined as a design variable. Although SIMP is a widely used structural topology optimization method, however the predefined penalty exponent of this method causes the premature convergence to local optima. In addition, SIMP method requires an additional filtering arrangement to avoid the checkerboarding and mesh dependency problem [Deaton and Grandhi 2014]. This arrangement limits the computational efficiency of the SIMP method [Deaton and Grandhi 2014].

Evolutionary Structural Optimization (ESO)

Evolutionary Structural Optimization (ESO) is another numerical topology optimization method which is incorporated into finite element analysis [Xie and Steven 1993]. The ESO method tries to obtain the optimum design of a structure by gradually removing the low stressed materials from the structure. The simplified version of ESO is described as follows:

- Step 1: Divide the design domain into a number of small elements using a fine mesh of finite elements;
- Step 2: Calculate the stress of each element by carrying out finite element analysis.
- Step 3: Determine the low stressed materials and remove from the design domain.
- Step 4: Increase the rejection ratio if a steady state is being reached.
- Step 5: Repeat Steps 2 to 4 until the desired optimum is obtained.

ESO is relatively a simple topology optimization method. However, ESO is highly inefficient, because it first obtains a large number of solutions by using a heuristic method and after that, a searching mechanism is initiated to search through this set of solutions to locate the desired optimal solution. In addition, its premature elements which have already been removed from the structure cannot be recovered. Hence, the amount of removed

materials needs to be kept as small as possible. Otherwise, an unoptimized solution will be obtained. According to [Zhou and Rozvany 2001], ESO method is unable to provide any mathematical proof that ensures the structure evolved to obtain an expected optimum.

Level-set Method

The level-set method was first introduced in [Wang et al. 2003] for structural topology optimization based on the level-set model proposed in [Osher and Sethian 1988]. In the level-set method, the design domain D is implicitly represented by a moving boundary $\partial\Omega$ which is embedded by the zero level set of the level set function $\Phi(x)$, as shown in Fig. 2.9. Throughout the optimization, the level set function can move up and down within a fixed Euler grid. This movement causes the surrounded boundary to experience a radical shape or topological change. During the course of optimization, the value of $\Phi(x)$ on the boundary always kept to zero. Although the level set method allows natural topological changes, however, this method cannot find a good topology when there is no minimized path that reaches it and thus it may get trapped in local optima.

2.5.2 Metaheuristic Methods

In the previous section, we provide a brief description of the classical structural optimization methods such as the SIMP [Bendsøe 1989], ESO [Xie and Steven 1993], and level-set [Wang et al. 2003] methods. Basically, the classical structural optimization methods that have been developed rely on the use of gradient information derived from the specific problem formulation. Hence, the final solutions obtained by these approaches for the same problem usually differ according to the different starting solutions. In order to overcome the disadvantages of gradient-based approaches to solve the structural optimization problems, nowadays metaheuristic approaches such as Genetic Algorithm [Deb and Gulati

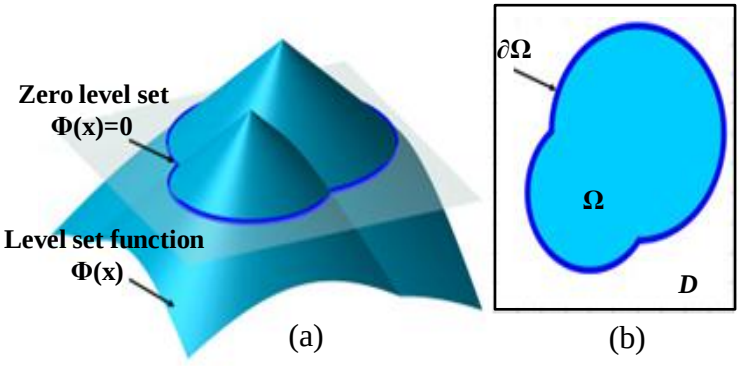


Figure 2.9: (a) Illustration (a) 3D level set surface and (b) its 2D boundary at zero level set [Wang et al. 2014].

2001, Tai and Akhtar 2005b, Li 2015], Ant Colony Optimization [Kaveh and Talatahari 2010, Gan et al. 2017], Binary Particle Swarm Optimizer [Perez and Behdinan 2007, Kaveh et al. 2014], Harmony Search [Degertekin 2008] methods are drawing a great attention to the researchers.

Metaheuristic approaches are stochastic global optimization methods. These approaches use the binary bits 1/0 to represent the solid/void material distribution, which represents a direct approach to handle structural topology optimization issues. Unlike gradient-based approaches, metaheuristic methods usually carry out a more exploratory global search at the start of the optimization and the results produced are less sensitive to initial starting positions. So far, most of above methods have been developed for obtaining a single optimal design solution despite the existence of multiple design solutions in a structural design domain [Deb and Gulati 2001, Miguel et al. 2013]. From the practical point of view, it is beneficial to find these solution for the several reasons as discussed in the introduction chapter.

2.6 Multimodal Optimization

Many real-world optimization problems including structural optimization problems can be characterized by multimodal fitness landscapes where multiple optimal (or close to optimal) solutions exist. A method that provides multiple solutions allows a decision maker to determine which one to eventually choose with respect to different conditions. If we want to obtain such optimal solutions by traditional metaheuristic algorithms, they need to be applied through several independent runs, and each time hoping to find a different optimal solution. Yet all solutions are not guaranteed to be different. An evolutionary algorithm (EA) can be used to locate multiple solutions by multimodal optimization algorithms which are commonly known as niching methods. Niching methods are well-suited optimization algorithms for finding multiple optimal solutions in a single optimization run [Goldberg and Richardson 1987]. As an example, an illustration of a run by such a niching method is presented in Fig. 2.10. In this figure, the top-most subfigure shows that a population of individuals take their initial positions in the different regions of the search space, the middle subfigure shows that the niching technique divides the whole population into subpopulations in order to form different niches where each of the niches represent a potential solution of the problem under consideration, and the lowermost subfigure shows that after N^{th} generation, the niching method found the different solutions within the same run.

2.6.1 Niching Methods

In literature, several niching methods have been proposed for the multimodal optimization problems. Classic niching methods include fitness sharing [Goldberg and Richardson

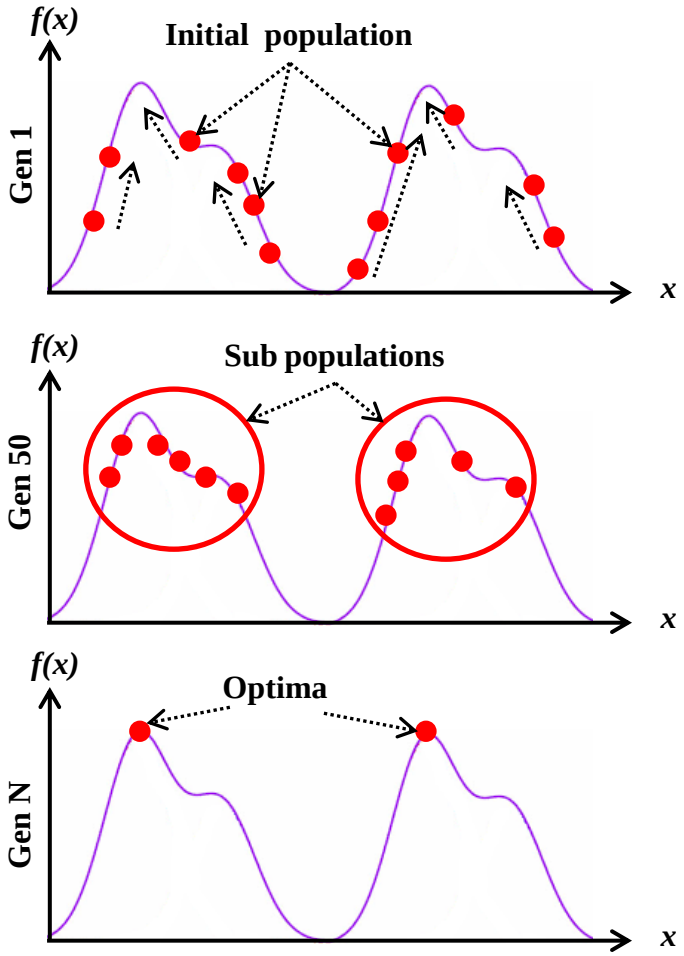


Figure 2.10: Illustration of the idea of locating multiple optimal solutions.

1987], clearing [Petrowski 1996], deterministic crowding [Mahfoud 1995], sequential niche [Beasley et al. 1993], speciation [Li et al. 2002], and speciation-based PSO (SPSO) [Li 2004]. The following sections contain a brief description of these popular niching methods.

Fitness Sharing

Literature shows that the fitness sharing is the most widely recognized niching method. It was proposed by Goldberg and Richardson in 1987 [Goldberg and Richardson 1987]. Inspired by the sharing concept observed in nature, it assumes that there is only limited and fixed resource available at each niche. Individuals in a niche must share their fitness value with its neighbours in that niche. This method considers the fitness of the landscape as a resource to be shared among the similar individuals of the population. Given by a similarity measure that is defined by distance d_{ij}, sharing radius δ_{share}, and sharing

function $f(d_{ij})$; the fitness sharing method can be described in the following:

- Step 1: Calculate the distance d_{ij} between two individuals i and j.
- Step 2: Calculate the sharing function $f(d_{ij})$ between two individuals by the following equation:

$$sh(d_{ij}) = \begin{cases} 1 - (\frac{d_{ij}}{\delta_{share}}) \text{ if } d_{ij} < \delta_{share}, \\ 0 \qquad\qquad \text{otherwise.} \end{cases} \tag{2.12}$$

- Step 3: Calculate the shared fitness of individual i by the following equation:

$$f_{share}(i) = \frac{f_i}{\sum_{j=1}^{popsize} sh(d_{ij})}, \tag{2.13}$$

  where f_i is the original fitness of the individual i and $popsize$ is the population size.

Fitness sharing method is widely used to identify and maintain the multiple optima in a single run. However, it has following drawbacks:

- Assumption of the niche radius δ_{share}. It assumes that the optima are far enough from each other with respect to the niche radius, which is predetermined for the particular problems and remains fixed during the optimization process.
- Fitness sharing uses $\mathcal{O}(popsize^2)$ time for the distance calculation between two individuals to get the shared fitness, which increases rapidly with the number of peaks and population size.

Crowding

Crowding method is another popular niching method [De Jong 1975]. Initially, this method was designed to preserve population diversity and prevent premature convergence of GA. In this method, an offspring is compared with a small random sample taken from the current population, and only the similar individual in the sample get replaced. An user-defined parameter namely crowding factor (CF) is often used to determine the size of the sample. The crowding niching method facilitates the growth of individuals around underpopulated regions of the solution space. In [Mahfoud 1995], it is mentioned that the crowding method is unable to maintain more than two peaks of multimodal fitness landscapes, as it failed to prevent genetic drift in many cases. To overcome this difficulty, the deterministic crowding method was proposed by Mahfoud in 1995 [Mahfoud 1995]. Deterministic crowding method does not depend on any predefined parameters like niche radius in fitness sharing method or crowding factor in crowding method. In this method, similar individuals compete for limited resources. Once reaching the capacity, each weak

individual is discarded from the population. The algorithm for deterministic crowding method is as follows:

Repeat until the stopping criteria are not satisfied.

- Step 1: Randomly select two parents p_1 and p_2 with replacement from the current population.
- Step 2: Generate two offspring c_1 and c_2 using the variation operators.
- Step 3: Evaluate the fitness value of offspring, $f(c_1)$ and $f(c_2)$, and calculate their distances to parents, i.e., $d(p_1, c_1)$, $d(p_1, c_2)$, $d(p_2, c_1)$, and $d(p_2, c_2)$.
- Step 4: Identify a close competition pair. If $[d(p_1, c_1) + d(p_2, c_2)] \leq [d(p_1, c_2) + d(p_2, c_1)]$, then the competition starts between $p_1 \leftrightarrow c_1$ and $p_2 \leftrightarrow c_2$. Otherwise, the competition will be in between $p_1 \leftrightarrow c_2$ and $p_2 \leftrightarrow c_1$.
- Step 5: Determine the winner. Individuals with higher fitness value win the competition and will stay in the population. Losers will be discarded.

Deterministic crowding method can discover peaks very fast but maintains them only for a few generations. On the other hand, deterministic rule promotes convergence but might suffer from premature convergence.

Clearing

The clearing method is very similar to fitness sharing but is based on the concept of limited resources of the environment [Petrowski 1996]. Clearing does not divide resources equally among all individuals of the subpopulation as in fitness sharing but supplies these resources only to the best-fit individual of the subpopulation. In practice, a species only accommodates a predefined number of individuals called the niche capacity K. Thus, clearing preserves the fitness of the best individuals of the species and eliminate the remaining individuals in the same species and re-initialize them from scratch. A simplified version of the clearing procedure is presented below in Algorithm 2.

Species-conserving Genetic Algorithm (SCGA)

The SCGA niching method is developed based on the concept of species [Li et al. 2002]. Basically, a species is formed by the individuals in a population which has similar characteristics and dominated by the best-fit individual, called the species seed. Briefly, a species S_i is centered upon its dominating individual i.e., the species seed $\mathbf{x}^*$, if for every individual $\mathbf{y} \in S_i$

$$\textbf{Distance}(\mathbf{x}^*, \mathbf{y}) < r_s, \tag{2.14}$$

Algorithm 2 Pseudocode for the clearing procedure.

Require: $\mathbf{P}$ and n ▷ Population $\mathbf{P}$ of n individuals
Require: r ▷ r is the niche radius
Require: K ▷ K is the capacity of each niche
Require: $nbWinners$ ▷ the number of winners in the subpopulation

```
SortFitness(P)
for i:=0 to n-1 do
    if Fitness(P_i) > 0 then
        nbWinners:=1;
        for j:=i+1 to n − 1 do
            if Fitness(P_i) > 0 and Distance(P_i,P_j)< r then
                if nbWinners < K then
                    nbWinners:=nbWinners+1;
                else
                    Fitness(P_j):=0.0;
                end if
            end if
        end for
    end if
end for
```

and

$$\textbf{Fitness}(\mathbf{y}) < \textbf{Fitness}(\mathbf{x}) \tag{2.15}$$

where r_s represents the species radius. For clarity, the Pseudocode for the SCGA niching method is demonstrated in Algorithm 3.

Speciation-based PSO (SPSO)

The speciation-based PSO (SPSO) [Li 2004] is a more speciation method coupled with the popular PSO algorithm. In SPSO, speciation involves the splitting of a single population into a number of subpopulations (**species**) according to their similarities measured by the Euclidean distance. The smaller the Euclidean distance between two population individuals, the more similar they are:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{k=1}^{d} (x_{i,k} - x_{j,k})^2}, \tag{2.16}$$

where, d represents the search dimensions, $\mathbf{x}_i$=$[x_{i1}, x_{i2}, \ldots, x_{id}]$ and $\mathbf{x}_j$=$[x_{j1}, x_{j2}, \ldots, x_{jd}]$ are the position of two population individuals from the PSO population.

The formation of species depends on a user-defined parameter r_s, which denotes the radius measured in Euclidean distance from the center of a species to its boundary. The center of a species, so-called **species seed**, is always the best-fit individual in the species. All the population individuals that fall within the r_s distance from the species seed are classified as being in the same species. Algorithm 4 summarizes the working steps for

Algorithm 3 Pseudocode for the SCGA niching method [Li et al. 2002].

```
1: t = 0;
2: Initialize G(t);
3: Evaluate G(t);
4: while not termination condition met do
5:     Determine species seeds X_s;
6:     Select G(t + 1);
7:     Crossover G(t + 1);
8:     Mutate G(t + 1);
9:     Evaluate G(t + 1);
10:    Conserve species from X_s in G(t + 1);
11:    t = t + 1;
12: end while
13: Identify global optima;
```

determining the species seeds from the whole population [Li 2004]. More specifically, this algorithm takes a list of all particles sorted in descending order of their fitness which is denoted by L_{sorted}. The species seed set S is initialized as an empty set. The particles in L_{sorted} are checked one by one against the species seeds found so far. If a particle does not reside within the species radius r_s of all the seeds of S, then this particle treats as a new seed and added to S.

The overall idea of SPSO is provided in the followings. In the beginning, SPSO divides the whole population into a number of species according to their similarities. The members of each species are assigned to their own dominating members called species seed. Once these species seeds are determined, the current generation is preserved by sending them into the next generation. In the following, the working steps of SPSO algorithm are provided.

- Step 1: Create a swarm of n number of particles.
- Step 2: Calculate the particle's fitness value along with setting their personal best position.
- Step 3: Sort the particles in descending order (according to their fitness value).
- Step 4: Determine species from the current population.
- Step 5: Members of each species are assigned to their own dominating members called species seeds (see Algorithm 4).
- Step 6: Update particle's velocity and position according to Eq. (2.5) and Eq. (2.7), respectively.
- Step7: Stop once the termination condition is fulfilled, otherwise go back to Step 2.

Algorithm 4 Pseudocode for determining the species seeds [Li 2004]

Input: L_{sorted}- a list of sorted particles
Output: S- a list of species seeds (dominating particles)
$S \leftarrow \emptyset$;
$found \leftarrow$ FALSE;
repeat
    Get best $p \in L_{sorted}$
    for *all* s $\in S$ **do**
        if $d(s,p) \leq r_s$ **then**
            $found \leftarrow$ TRUE;
            break;
        end if
    end for
    if *not found* **then**
        S.Add(p);
    end if
until *not reaching the end of* L_{sorted}

2.6.2 Niching Applied to Multimodal Structural Optimization

The existing niching methods including the fitness sharing and SPSO niching methods have been developed for the continuous multimodal optimization problems. These niching methods can be modified for discrete optimization and subsequently for handling multimodal combinatorial like topology optimization problems. Following this line of thinking, an artificial immune system algorithm based fitness sharing method is developed for multimodal structural optimization problems [Luh and Chueh 2004]. Due to the high computational cost and implementation complexity of this method, the same authors [Luh and Chueh 2004] developed another more efficient method based on fitness sharing and BPSO[Luh and Lin 2011]. Basically, it is a two-stage approach which assumes that structural topology and size variables are linearly separable. Given this assumption, this method finds different topologies using a BPSO method in the first stage by considering an equal cross-sectional area for each member of a ground structure. In the second stage, the size (cross-sectional areas) of the found topologies are optimized by the attractive and repulsive particle swarm optimization (ARPSO). It is obvious that with this approach, the optimal designs may not be attainable since these variables are often not linearly separable in practice [Deb and Gulati 2001]. For clarity, the pseudo-code for BPSO and ARPSO algorithms are provided in Algorithm 5.

In [Li 2015], an improved species-conserving genetic algorithm is introduced for the structural topology optimization problems. For clarity, the pseudo-code for the improved species-conserving genetic algorithm is provided in Algorithm 6. Briefly, this method first applied niching to find the multiple size solutions of a structural problem. After that, the truss topologies are obtained by means of discarding unnecessary members (the members

Algorithm 5 Pseudocode for the BPSO and ARPSO algorithms [Luh and Lin 2011].

//initialization
T_{BPSO}=Number of iteration of BPSO
T_{ARPSO}=Number of iteration of ARPSO
procedure BPSO-TOPOLOGY
    Set parameters Np_{BPSO}, c_1, c_2, β, [$\mathbf{V}^{max}$,$\mathbf{V}^{min}$], [$\mathbf{X}_g^{max}$,$\mathbf{X}_g^{min}$], δ_s, N_{memory}
    Set $k = 0$;
    Randomly initialize positions and velocities of particles ($\mathbf{X}_{g,i}^0$ and $\mathbf{V}_{g,i}^0$);
    Transform $\mathbf{X}_{g,i}^0$ to $\mathbf{X}_{p,i}^0$ using sigmoid function S($\mathbf{V}_{g,i}^0$);
    while $k < T_{BPSO}$ **do**
        for each particle i in particle swarm **do**
            Calculate the fitness $f^i(\mathbf{A},\xi)$ and sharing fitness $f^i_{share}(\mathbf{A},\xi)$ for particle i;
            Update local best $\mathbf{P}^k_{best,i}$ and global best $\mathbf{G}^k_{best,i}$;
            Renew particle memory;
            Update particle swarm $\mathbf{X}_{g,i}^{k+1}$, $\mathbf{V}_{g,i}^{k+1}$;
            Transform genotypic particle $\mathbf{X}_{g,i}^{k+1}$ to phenotypic particle $\mathbf{X}_{p,i}^{k+1}$;
        end for
        $k = k + 1$;
    end while
end procedure
procedure ARPSO-TRUSS
    Set parameters Np_M, c_1, c_2, β, [$\mathbf{V}^{max}$,$\mathbf{V}^{min}$], [$\mathbf{X}_g^{max}$,$\mathbf{X}_g^{min}$], d_{low}, d_{high}
    Set $l = 1$;
    while $l \leq N_{memory}$ **do**
        Set $k = 0$
        Randomly initialize positions and velocities of particles ($\mathbf{X}_i^0$ and $\mathbf{V}_i^0$)
        while $k < T_{AESO}$ **do**
            for each particle i in particle swarm **do**
                Calculate the fitness $f^i(\mathbf{A},\xi)$ for particle i;
                Update local best $\mathbf{P}^k_{best,i}$ and global best $\mathbf{G}^k_{best,i}$;
                Update particle swarm $\mathbf{X}_i^{k+1}$, $\mathbf{V}_i^{k+1}$;
            end for
            $k = k + 1$;
        end while
        $l = l + 1$;
    end while
end procedure

whose cross-sectional areas fall below a given threshold value) for the found size solutions. It is obvious that this method may not obtain the optimal design solution because the topology solutions are highly dependent on the found size solutions of the problem under consideration.

Apart from the above methods, a few other research works have taken initiative for developing non-niching methods to find multiple solutions of structural design problems. For example, Deb and Gulati [Deb and Gulati 2001] proposed a genetic algorithm to show that there exist multiple different topologies for the structural design problems with almost equal overall weight. Another GA based structural optimization method is presented in [Balling et al. 2006] for multimodal structural topology optimization. In [Miguel et al.

Algorithm 6 Pseudocode for the improved species-conserving genetic algorithm [Li 2015].

```
1: t = 0;
2: Initialize G(t);
3: Evaluate G(t);
4: while not termination condition met do
5:     Determine species seeds X_s;
6:     Select G(t + 1);
7:     Crossover G(t + 1);
8:     Mutate G(t + 1);
9:     Evaluate G(t + 1);
10:    Conserve species from X_s in G(t + 1);
11:    Species Mutation;
12:    t = t + 1;
13: end while
14: Identify global optima;
```

2013], a single-stage firefly-based algorithm is proposed for the same purpose. Note that these methods have been developed for the structural topology optimization only where the size variables of the considered problems assumed to be fixed during the optimization.

It can be summarized that the existing structural optimization methods have the difficulties of finding good quality design solutions for structural design problems due to their inherent limitations that are mentioned above. In addition, these methods are not specifically designed for finding multiple design solutions. Hence, it is necessary to overcome the limitations of these existing methods to find the structural topology and size solutions in a simultaneous manner.

2.7 Multimodal Benchmark Functions

Many real-world discrete optimization problems including the 0-1 knapsack problem [Pisinger 1995], traveling salesman problem [Tsai et al. 2004], vehicle routing problem [Gendreau et al. 1994a], bin packing problem [Martello and Vigo 1998], and structural optimization problem [Christensen and Klarbring 2009], have multiple good solutions and often require multimodal optimization methods to locate these optimal solutions.

In the field of evolutionary computation (EC), it is a common practice that the weakness and strengths of a newly developed niching algorithms can be evaluated over artificial test problems before applying them to real-world problems. In this context, several attempts have been made to design multimodal test problems [Deb 1989, Mahfoud 1995, Li et al. 2002] for evaluating the performance of multimodal optimization methods such as niching methods [Holland 1992a, Goldberg and Richardson 1987, Mahfoud 1995, Petrowski 1996]. However, most of these test problems are limited to low-dimensional continuous solution space (see Fig. 2.11), and they cannot be easily tuned. This makes it difficult to evaluate a niching method's. To overcome these drawbacks, efforts have been made to design frameworks for generating controllable challenging multimodal benchmark

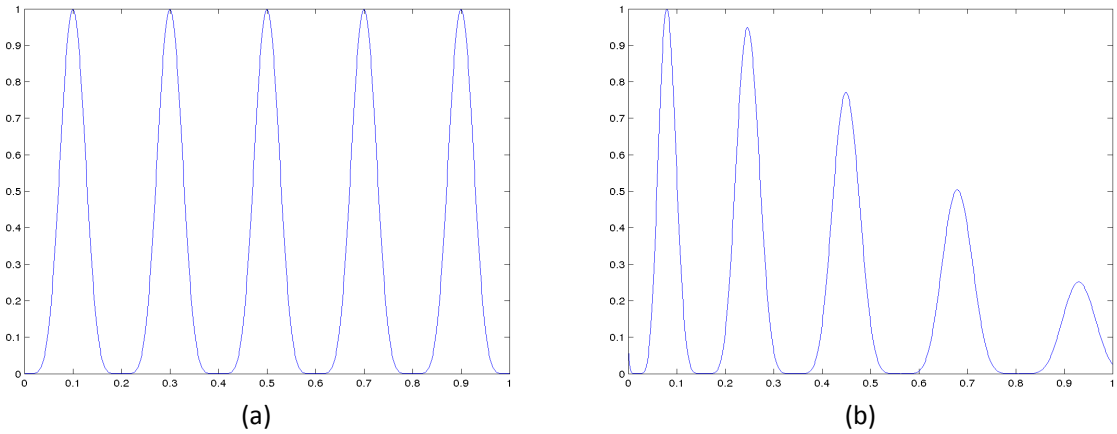


Figure 2.11: Illustration of the (a) test function having equal maxima and (b) test function having uneven decreasing maxima [Mahfoud 1995].

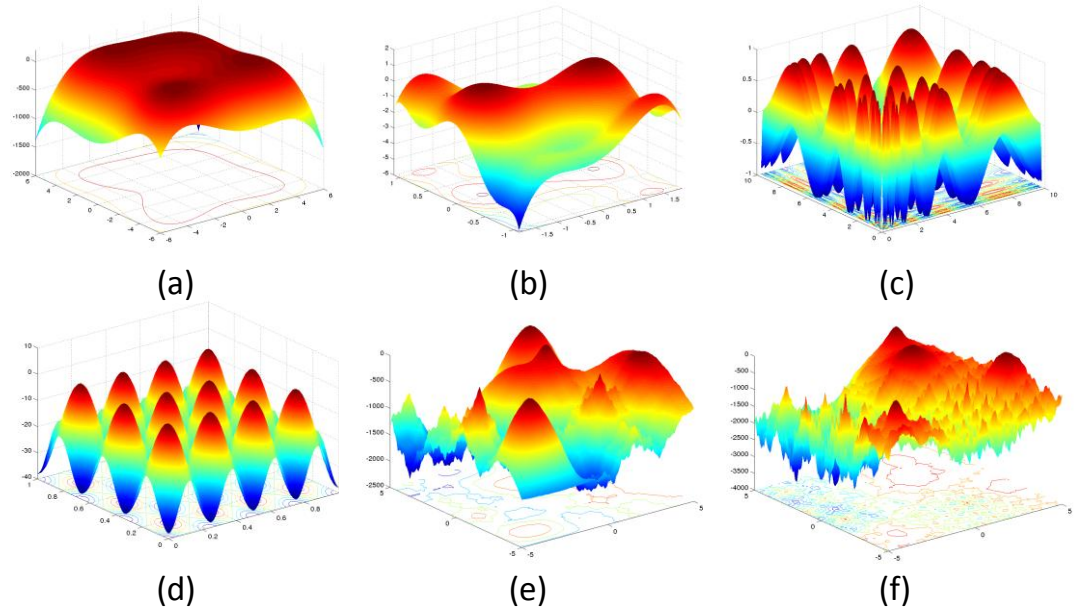


Figure 2.12: Illustration of the (a) Himmelblau function, (b) Six-Hump camel back function, (c) Vincent function, (d) Modified Rastrigin function, (e) Composition function 1, and (f) Composition function 2 [Rönkkönen et al. 2011, Li et al. 2013].

problems (see Fig. 2.12) [Rönkkönen et al. 2008; 2011, Li et al. 2013]. Nevertheless, all these benchmark problems have been designed for the continuous optimization problems. There are no well-defined test problems for the discrete multimodal optimization problems in literature, which can be used to evaluate the weakness and strengths of a developed discrete niching algorithm.

2.8 Bilevel Optimization

Multi-level and bilevel optimization have been widely studied in the field of mathematical programming [Bard 1998, Colson et al. 2007, Dempe 2002]. Multi-level and bilevel optimization have emerged as an important research area for progress in handling many real-life problems from a wide range of domain areas. The formulation of the multi-level programming was first introduced in 1973 by J. Bracken and J. McGill [Bracken and McGill 1973]. Multi-level optimization is developed for distributed planning problems in a hierarchical organization where more than one decision makers are involved. The decision makers have taken their decisions in a sequential manner without any cooperation. The multi-level can be viewed as a hierarchy of planners where each planner is independent to control a subset of decision variables.

A bilevel optimization problem can be viewed as a multi-level optimization problem with two levels. Particularly, in a bilevel optimization problem, two different levels of optimization take place, with one level (i.e., the lower level) of optimization being nested within the other (i.e., the upper level), as illustrated in Fig. 2.13. For a bilevel problem, if the upper level optimizer wants to optimize its objective, then it needs to obtain an optimal response of the lower level optimizer [Bard 1998]. For the upper level objective function F and lower level objective function f, the bilevel optimization problem can be expressed as:

$$\begin{aligned} \min_{\vec{x_u} \in X_u, \vec{x_l} \in X_l} \quad & F(\vec{x_u}, \vec{x_l}) \\ s.t. \quad & \vec{x_l} \in \operatorname*{argmin}_{\vec{x_l} \in X_l} \{ f(\vec{x_u}, \vec{x_l}) : g_j(\vec{x_u}, \vec{x_l}) \leq 0 \} \\ & G_k(\vec{x_u}, \vec{x_l}) \leq 0, \end{aligned} \tag{2.17}$$

where $j = 1, 2, \ldots, J$ and $k = 1, 2, \ldots, K$, G_k represents the upper level constraints and g_j represents the lower level constraints, respectively. In this formulation, the upper level objective function evaluates the performance of the lower level objective function through $f(\vec{x_u},\vec{x_l})$, which is obtained by solving the lower level variable $\vec{x_l}$ for fixed $\vec{x_u}$.

The idea of bilevel optimization has been applied to various real-world optimization problems including toll setting problem [Kalashnikov et al. 2016], chemical industry [Halter and Mostaghim 2006], defense applications [Aksen and Aras 2012], and inverse optimal control [Mombaur et al. 2010]. In addition to these, the idea of bilevel optimization is also applied to the structural optimization problem. For example, in [Herskovits et al. 2000], the bilevel formulation is used for the structural shape optimization. In [Christiansen et al. 2001], bilevel programming approach is used for the structural size optimization. The same idea is adopted in [Kočvara 1997] for the structural topology optimization problem. Recently, the bilevel formulation is employed in [Ahrari and Deb 2016] for the layout optimization of the structural design problems. It can be observed that these bilevel

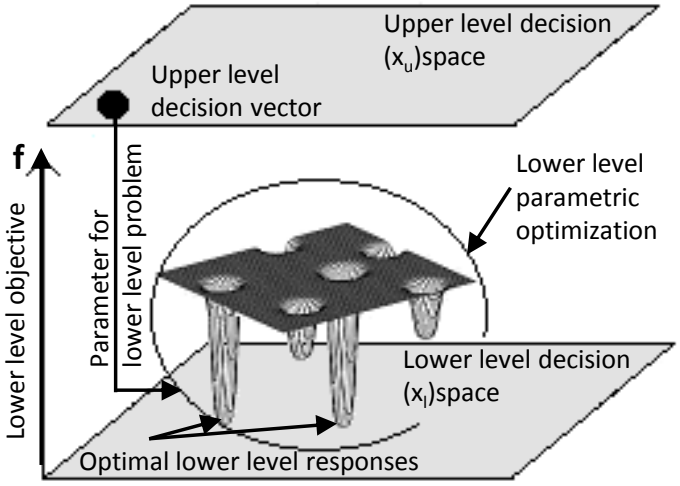


Figure 2.13: Illustration of the relationship between the upper level and lower level search spaces of a general bilevel problem.

formulations have been successfully applied for either the topology or size optimization for structural design problems. Nevertheless, the bilevel concept has been applied for the simultaneous structural topology and size optimization which can be found in the literature [Colson et al. 2007, Sinha et al. 2017].

2.9 Chapter Summary

This chapter discussed the background materials of this thesis. Firstly, an overview of general optimization and its popular optimization techniques are provided. Secondly, an overview of structural optimization and its well-known optimization methods are introduced. Thirdly, the widely used multimodal optimization methods (i.e., niching methods) and their applicability to multimodal structural optimization problems are discussed. Fourthly, an overview of multimodal benchmark functions is provided. Finally, the basic idea of bilevel optimization and its applicability to the structural optimization are discussed.

In this chapter, we identified five research gaps. Firstly, most existing discrete metaheuristic methods have the difficulties of maintaining a good balance between exploration and exploitation which need to be addressed. Secondly, it is necessary to develop a discrete niching method to handle the structural topology like discrete optimization problem, since there is no such well-known niching method exists in the literature. Thirdly, it is necessary to design discrete multimodal benchmark functions for identifying the weaknesses and strengths of discrete niching methods. Fourthly, it is necessary to formulate the structural optimization problem as a bilevel optimization problem in order to facili-

tate the simultaneous topology and size optimization. Finally, a bilevel niching method is necessary to develop for the bilevel structural optimization problems so that multiple topologies and their size solutions can be found in a single optimization run. The following chapters provide the research contributions towards addressing these identified research gaps.

CHAPTER 3

Designing Multimodal 0-1 Knapsack Test Problems

3.1 Introduction

Test suites are important for the performance evaluation of multimodal optimization methods. In the last two decades, several attempts have been made to design test suites for generating multimodal test problems. However, these research works have been limited to continuous optimization problems. In practice, many optimization problems are combinatorial and multimodal by nature. To facilitate analysis of the properties of multimodal combinatorial optimization problems, a well-defined test suite is required. Literature shows that there is no such test suite exists which can be used for this purpose. In this chapter, we choose the 0-1 knapsack problem as a representative NP-hard combinatorial optimization problem for generating a multimodal combinatorial test problem suite. We propose a framework for generating a diverse set of challenging multimodal test instances using the 0-1 knapsack problem. The proposed framework is flexible allowing a user to have a better control on the settings of the multimodal test instances such as dimensionality, distribution of the optima, the number of optima, and fitness of the optima. To demonstrate the generalization capability of this framework, we propose 14 standard multimodal test instances. We further illustrate the usefulness of this multimodal test function suite using a binary particle swarm optimization based niching method.

The rest of the chapter is organized as follows. In Section 3.2, the research motivation of this chapter is presented. Section 3.3 describes the background and a brief survey of existing research works on the 0-1 knapsack problem. Section 3.4 describes the specifying procedure of multimodal 0-1 knapsack parameters. Section 3.5 demonstrates the software framework for generating multimodal 0-1 knapsack instances. Section 3.6 demonstrate the proposed multimodal test instances. We present a binary niching method followed

by some experimental studies in Section 3.7. The concluding remarks of this chapter are provided in Section 3.8.

3.2 Motivation

Many real-world problems belong to the family of discrete/combinatorial optimization problems and they are multimodal by nature[1] [Pisinger 1995, He et al. 2016, Gendreau et al. 1994a, Laskari et al. 2005, Tsai et al. 2004]. To facilitate analyzing the properties of this kind of problems, a well-defined benchmark is needed. Literature review shows that existing multimodal benchmark suites are largely artificial continuous functions [Goldberg 1989, Mahfoud 1995, Rönkkönen et al. 2008, Mirjalili and Lewis 2015]. There is no well-defined benchmark for multimodal combinatorial optimization. Although there exist many multimodal optimization methods [Goldberg 1989, Mahfoud 1995, Petrowski 1996, Parrott and Li 2006] developed for solving the continuous optimization problems, their behaviour and performance cannot be guaranteed when applying them to combinatorial optimization problems. Therefore, it is necessary to generate well-defined benchmarks specifically designed for multimodal combinatorial optimization. In this chapter, we choose the 0-1 knapsack problem as a representative combinatorial optimization problem due to its simplicity and a wide range of applications in practice [Mavrotas et al. 2008, Wscher et al. 2007, Higgins et al. 2008].

The 0-1 knapsack problem has been studied for more than a century, with early works dating as far back as 1896 [Mathews 1896]. It is one of the classical NP-hard constrained combinatorial optimization problems [Williamson and Shmoys 2011]. Building on the early studies of 0-1 knapsack problem, many research works have been conducted to design methods for generating hard 0-1 knapsack test instances [Martello and Toth 1979, Chvátal 1980, Pisinger 2005]. However, to the best of our knowledge, none of these methods can generate multimodal test instances with controllable dimensionality, the number of optima, fitness of the optima, and distribution of optima of the 0-1 knapsack problem.

3.3 Related Work

In this section, we provide a brief survey of existing research works on the classic 0-1 knapsack problem. The 0-1 knapsack problem is a classical NP-hard combinatorial optimization problem. It can be expressed as follows. Suppose we have a set of n items $\mathbf{i}=\{i_1, i_2, \ldots, i_n\}$ and each item has a weight w_i and a profit p_i. The goal is to select a subset of items in order to maximize the total profit such that the total weight of the selected items does not exceed the knapsack capacity C. To model this problem, x_i is

[1] The more generic background information on discrete/combinatorial optimization and multimodality can be found in Chapter 2.

introduced as a decision variable for each item. If the i-th item is selected then the value of x_i is set to 1, otherwise its value is set to 0. Formally, the 0-1 knapsack problem can be expressed as:

$$\begin{aligned} \text{Maximize} \sum_{i=1}^{n} p_i x_i \\ \text{s.t.} \sum_{i=1}^{n} w_i x_i \leq C, \\ x_i \in \{0,1\}. \end{aligned} \tag{3.1}$$

The 0-1 knapsack problem has been studied in the last few decades because of its theoretical interest and its wide applicability in operations research and engineering. Several attempts have been made to design frameworks for generating 0-1 knapsack test instances [Martello and Toth 1979, Chvátal 1980, Balas and Zemel 1980, Chung et al. 1988, Pisinger 2005, Bansal and Deep 2012, Zou et al. 2011b, Bhattacharjee and Sarmah 2014] in order to facilitate the performance evaluations of classical [Martello and Toth 1997, Monaci et al. 2013] and evolutionary 0-1 knapsack algorithms [Ezziane 2002, Bansal and Deep 2012, Zou et al. 2011b]. In general, there are two groups of instances for evaluating the performance of the algorithms, *deterministic* instances and *non-deterministic* instances.

The instances in the first group are easier, because their optimal solutions can be found by exact methods due to the low dimensionality of the instances [Bansal and Deep 2012, Zou et al. 2011b, Bhattacharjee and Sarmah 2014]. For these test instances, there is no generation procedure available for specifying the 0-1 knapsack parameters. In addition, these instances are not scalable in terms of the number of dimensions, nor do they guarantee to contain multiple optima in the search space.

In contrast, the instances of the second group are the hardest test instances ever designed for the 0-1 knapsack problem [Martello and Toth 1979, Chvátal 1980, Balas and Zemel 1980, Chung et al. 1988, Pisinger 2005]. For this group, there are mainly five different types of test instances namely the *uncorrelated*, *weakly correlated*, *strongly correlated*, *inversely strongly correlated*, and *subset sum* instances which are defined by taking into account the correlation between the profits and weights of items [Pisinger 2005]. It is reported in [Pisinger 1995] that the strongly correlated and subset sum instances are difficult to solve than the uncorrelated and weakly correlated instances. The generation procedure of these five types of test instances are provided below:

- *Uncorrelated instances (UCI)*: For this type of test instances, the profit of each item is independent from its weight. In practice, the weight and profit of each item are chosen randomly from a certain interval, e.g., [1,R].

- *Weakly correlated instances (WCI)*: For this type, the weight of each item is randomly chosen from [1,R]. Next the profit (p_i) of an item (i) is selected from the

interval $[w_i - R/10\ ,\ w_i + R/10]$ subject to $p_i \geq 1$, where w_i is the weight of the item.

- *Strongly correlated instances (SCI)*: For these strongly correlated instances, the weight of an item is also randomly chosen from [1,R]. After that, the profit (p_i) of an item (i) of the strongly correlated instances is chosen as $p_i = w_i + R/10$.
- *Inversely strongly correlated instance (ISCI)*: For this type, first, the profit of an item i is randomly chosen from [1,R] and then the weight of the same item is selected as: $w_i = p_i + R/10$.
- *Subset sum instances (SSI)*: For this type of instances, the weight (w_i) of the i-th item is first chosen randomly from the interval [1,R], then assigned to the profit of the same item as $p_i = w_i$.

For each instance type, the knapsack capacity C is usually set to a certain percentage of the sum of weights of the items. Mathematically, the knapsack capacity is chosen as:

$$C = S \times \sum_{i=1}^{n} w_i, \tag{3.2}$$

where S is a variable which can be chosen from a real number between 0.1 and 0.9.

The generation procedure of the non-deterministic instance group can produce a diverse set of 0-1 knapsack instances. However, this procedure cannot guarantee multiple global optimal solutions to be produced for the 0-1 knapsack problem. This is because of the random specifications for the set of weights ($\mathbf{w}$) and the set of profits ($\mathbf{p}$) of a test instance, which is highly unlikely to produce optimal solutions of equal fitness values. This motivates us to propose a new generation procedure which will guarantee the multiple global optima in the search space of 0-1 knapsack instances. In addition, it will allow controlling various properties of the 0-1 knapsack instances such as the number of optima, fitness of the optima, location of the optima, and distribution of optima.

3.4 New Procedure for Generating Multimodal 0-1 Knapsack Instances

The following sections will describe how to specify the set of weights, the set of profits, and knapsack capacity in order to generate a multimodal test instance for the 0-1 knapsack problem, in particular the *strongly correlated* and *subset sum* multimodal 0-1 knapsack test instances, because they are much harder and have more interesting properties than the *uncorrelated* and *weakly correlated* instances [Pisinger 1995].

3.4.1 Weights

A commonly used technique to specify the set of weights $\mathbf{w}=\{w_1, w_2, \ldots, w_{(n-1)}, w_n\}$ randomly in a certain interval, e.g. $[1, r]$, where r is a positive integer number [Martello and Toth 1979, Chvátal 1980, Balas and Zemel 1980, Chung et al. 1988, Pisinger 2005]. In contrast, this study specifies the set of weights $\mathbf{w}$ with the pernicious numbers [Sloane 2017] to produce the multiple optima in the search space of a 0-1 knapsack instance. By definition, a number m is pernicious, if it contains a prime number of ones in its binary representation. For example, $14 = (1110)_2$ is pernicious since it contains 3 ones and 3 is a prime number. Likewise, the numbers 3, 5, 6, 7, 9, 10, 11, 12, 13, 17, 18, 19, 20, 21, 22, 24, 25, 26, and 28 are also pernicious. So far, the key idea of specify the elements of $\mathbf{w}$ is that at first a certain amount (equal to the number of items) of pernicious numbers is generated, then these numbers are randomly assigned to the elements of $\mathbf{w}$. According to this, an example of such a set of weights for a 10-dimensional 0-1 knapsack instance can be $\mathbf{w}=\{12, 7, 5, 13, 11, 10, 3, 9, 6, 17\}$.

3.4.2 Profits

In Section 3.3, five different types of profits can be found for generating the instances from the second group. It can be observed that these profits have been specified according to the certain correlations between the weights and profits of the items. In this chapter, two out of five existing correlations are used to specify the profits of multimodal test instances. However, the procedures for generating these profits (as described in Section 3.3) are modified in the following ways:

- *Strongly correlated profit (SCP)*: In this case, the profit of an item is strongly correlated with its weight plus certain shift [Pisinger 2005]. Following this, the strongly correlated profit ($p_i \in \mathbf{p}$) for the i-th item is chosen as: $p_i=w_i + (w_i \times r)$, where r is a positive integer number.

- *Subset sum profit (SSP)*: In this case, the profit of an item is a linear function of its weight. According to this, the set of subset sum profits is specified as: $\mathbf{p} = \mathbf{w} * r$.

3.4.3 Knapsack Capacity

In general, the existing frameworks assign the knapsack capacity by a certain percentage of the sum of the item weights [Martello and Toth 1979, Chvátal 1980, Balas and Zemel 1980, Chung et al. 1988, Pisinger 2005]. Therefore, these frameworks cannot guarantee multiple optima in the search space of a generated 0-1 knapsack instance. To overcome this, an investigation is made on the relationship between the weight, profit, and knapsack capacity of the 0-1 knapsack problem to identify the key factors influencing the generation of multimodal test instances. This investigation discovers that if a specific value is chosen

for knapsack capacity C of an instance to satisfy the following criteria, then multiple optimal solutions can be produced for that instance:

$$C = w_i, and\ w_i \geq 9, \tag{3.3}$$

where $w_i \in \mathbf{w}$ is the weight of the i-th item. Note that if the value of $C < 9$, then the generated instances will not contain multiple optima with the same fitness value.

3.5 The Proposed Framework for Multimodal 0-1 Knapsack Instances

In this chapter, the above generation procedure is referred to as the multimodal test instances generation framework. At this moment, this framework can produce the *strongly correlated* and *subset sum* multimodal 0-1 knapsack test instances. However, the procedure is easily modifiable for any types of multimodal test instances. For example, the new correlations between the profits and weights of the items can be applied and adopt them in this framework to generate different types of multimodal 0-1 knapsack instances.

For convenience, a software tool based on the idea of the new generation procedure is developed. Fig. 3.1 shows the user interface of this software tool which can be downloaded from here[2].

3.5.1 Examples of Generated Multimodal Instances

To generate a *SCMI* type multimodal 0-1 knapsack instance, for example, a user can enter the following generator parameter values: n=5, r=5, and **P**=SCP via the software interface (see Fig. 3.1). With this setting, the software tool produces the following *SCMI* instance characterized by: **w**={7, 3, 5, 6, 9}, **p**={42, 18, 30, 36, 54}, and C=3 to 9. This instance is referred to as *Instance*-1. To produce multiple global optima, the value for C should be chosen ≥ 9 which is already mentioned in Eq. (3.3). Otherwise, this framework produces a single global optima in the search of the generated instances. For example, let the weight of the first item is chosen as the knapsack capacity i.e., C=7. In this case, the search space of *Instance*-1 contains four optimal solutions according to Eq. (3.1). The four optimal solutions are "10000", "01000", "00100", and "00010", respectively. It can be observed that the profit of the global optimum "10000" is 42 which is better than the profit of three other optimums i.e., 18, 30, 36, respectively. This shows that if the value smaller than 9 is chosen for C, then a single optimum is produced in the search space of *Instance*-1.

It is worth to mention that the optima of the multimodal 0-1 knapsack instances of this study obey the following statement. The statement is: If $C = w_i$ and $w_i \geq 9$, then a

[2] https://drive.google.com/uc?export=download&id=0B7T6uXAYMncsUTJJclZ0W\DBTZVk

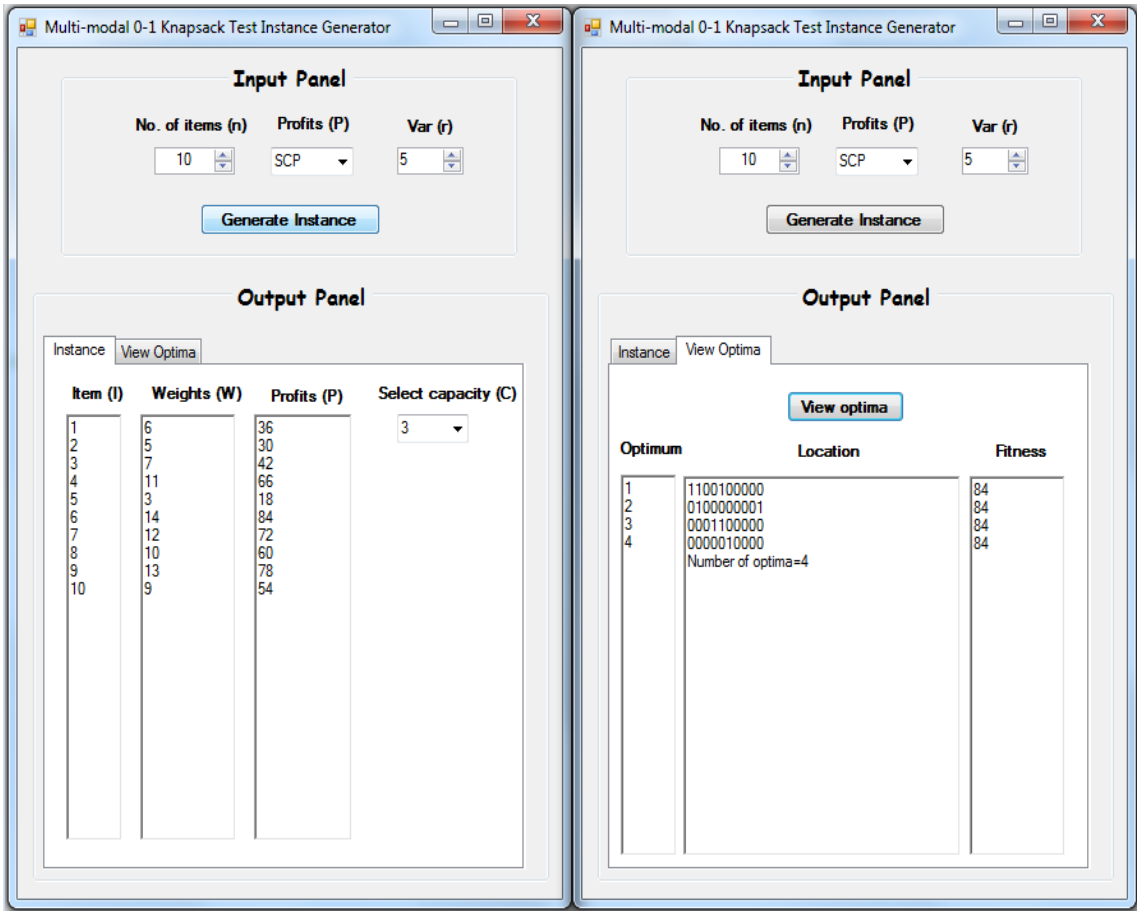


Figure 3.1: The user interface of the multimodal 0-1 knapsack test instances generator.

solution is globally optimal if and only if $\sum_{i=1}^{n} w_i = C$. An explanation of this statement is given below.

Let the knapsack capacity for the *Intance*-1 be C=9. In this case, the search space of this instance contains four optimal solutions "10000", "01100", "01010", and "00001", respectively, as shown in Fig. 3.2. Each of them are locally optimal, because all of their neighboring solutions are either worse (in terms of profit value) than it or infeasible i.e., $\sum_{i=1}^{n} w_i > C$. According to Eq. (3.1), the fitness value of these optimal solutions are 42, 48, 54, and 54, respectively. It is clear that the solutions "01010" and "00001" are the global optima, as their fitness values are higher than the fitness value of two other solutions. It can be found that with C=9, these two global optima fulfilled the condition $\sum_{i=1}^{n} w_i = C$, but two other optima did not fulfill the same given condition.

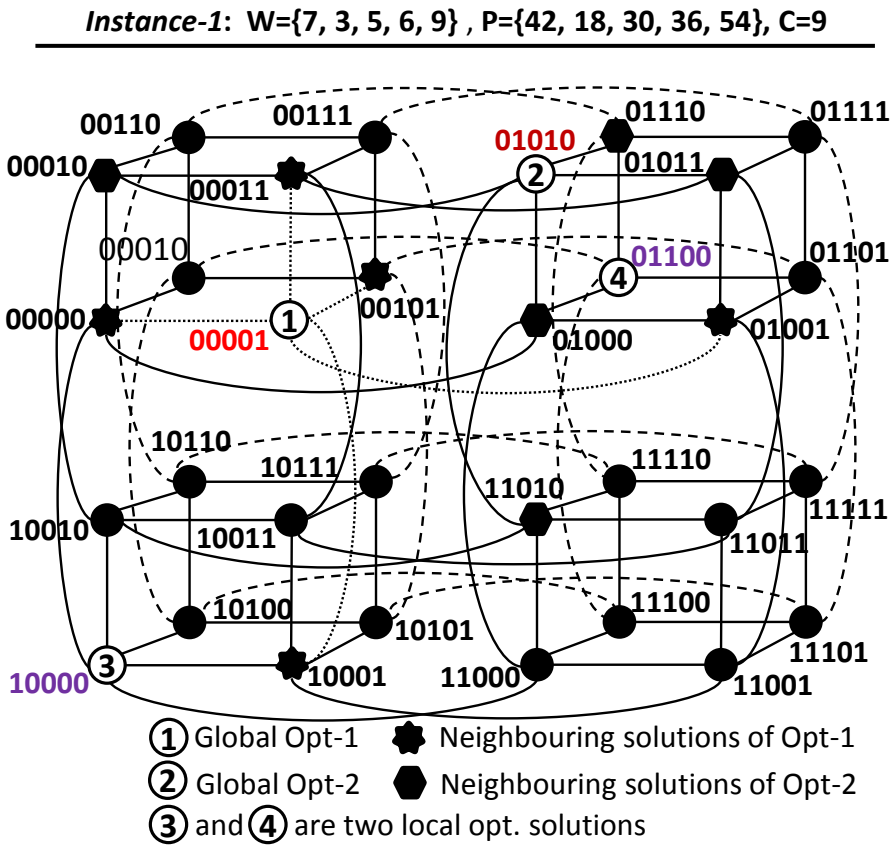


Figure 3.2: The search space of Instance-1 which contains two global optima with the same fitness value.

3.5.2 Optima of Generated Multimodal Instances

According to above description, an optimal solution is guaranteed by selecting a subset of items for which the accumulated weight is the same as the knapsack capacity C. Basically, the purpose of such arrangement is to know in advance that in a generated instance, how many optima exist, what is their fitness value, and what are the locations of the optima in the search space. For the demonstration purpose, a new multimodal 0-1 knapsack instance namely *Instance*-2 is generated by using the following settings n=20, r=5, and **P**=SCP. This instance is characterized by: **w**={14, 21, 5, 26, 24, 6, 19, 20, 17, 28, 12, 25, 9, 22, 3, 7, 11, 18, 13, 10}, **p**={84, 126, 30, 156, 144, 36, 114, 120, 102, 168, 72, 150, 54, 132, 18, 42, 66, 108, 78, 60}, and C=3 to 28.

To know the number of optima, we choose the knapsack capacity $C=w_{20}=10$ of *Instance*-2 based on Eq. (3.3). In this case, the search space of this instance contains only two global optima which follow the rule provided in the previous equation. The first one can be obtained by selecting only one item (the n-th item) of which the weight is 10 ($=C$) and the second one can be obtained by selecting only two items (15-th and 16-th

item) of which the weights are 3 and 7 (3+7=C), respectively. Now, we can compute the locations of these two optima by denoting the selected items of these optima as 1 and unselected items as 0. According to this, the locations of these two optima are "00000000000000000001" and "00000000000000110000", respectively. Finally, the fitness value of these two optima can be computed in the following way. The profit of the item whose weight is chosen as the knapsack capacity is the fitness value (60 in this case) of the optima of *Instance*-2. For any multimodal instances of this study, the fitness value of the optima can be computed in the similar way. For example, if C=w_{10}=28, then the fitness value of all the optima of this instance is the profit p_{10} of the 10-th item i.e., 168.

3.5.3 Controllable Properties of Multimodal Instances

The properties of the multimodal test instances can be controlled by the variables n, r, and C used in the generation procedure of multimodal 0-1 knapsack instances, where n determines the number of dimensions, C controls the number of optima, r controls the relative fitness of the optima of a multimodal test instance.

Dimensionality (Dim)

Since the 0-1 knapsack is NP-hard, the dimensionality of a test instance has a significant impact on the performance of multimodal optimization algorithms. Therefore, the proposed framework allows the user to control the dimensionality of the test instances in order to generate instances with various difficulties. The user can control the dimensionality of the test instances by tuning the value of n via the software tool.

Number of Optima

In multimodal optimization, the number of optimal solutions is an important factor to evaluate the performance of an optimization method. We can assess the ability of multimodal optimization methods in terms of the ability to locate multiple global optima simultaneously [Rönkkönen et al. 2011]. The user can control the number of optimal solutions of a particular test instance by tuning the value of knapsack capacity C. For example, *Instance*-3 is generated by setting n=20, r=5, and **P**=SCP via the software framework which is characterized by: **w**={20, 7, 25, 24, 14, 12, 19, 10, 28, 3, 11, 17, 18, 26, 13, 21, 22, 6, 5, 9}, **p**={120, 42, 150, 144, 84, 72, 114, 60, 168, 18, 66, 102, 108, 156, 78, 126, 132, 36, 30, 54}, and C=3 to 28. For a smaller value of C=12, the search space of *Instance*-3 contains only three optima of the fitness value 72, as demonstrated in Table 3.1. However, if the value of C increases to 17 and 22, then the same search space contains 6 different optima of the fitness value 102 and 12 different optima of the fitness value 132, respectively, as demonstrated in Table 3.1.

Table 3.1: The optima of *Instance*-2 for the different value of knapsack capacity C.

C	# Optima	Loc. of the optima	Weight	Profit
12	3	01000000000000000010	12	72
		00000100000000000000	12	72
		00000000010000000001	12	72
17	6	01000001000000000000	17	102
		00001000010000000000	17	102
		00000100000000000010	17	102
		00000000010000000011	17	102
		00000000001000000100	17	102
		00000000000100000000	17	102
22	12	01000100010000000000	22	132
		01000001000000000010	22	132
		01000000000000000101	22	132
		00001000010000000010	22	132
		00000101000000000000	22	132
		00000010010000000000	22	132
		00000001010000000001	22	132
		00000000010000100100	22	132
		00000000001000000110	22	132
		00000000000100000010	22	132
		00000000000000100001	22	132
		00000000000000001000	22	132

Relative Fitness of Optima

The distance between the fitness of the optima (local and global) is important when the purpose is not only identifying the global optima, but also the reasonably good local optima of multimodal optimization problems. In this regard, the proposed framework can be used to change the fitness of local and global optima. To do this, the user should choose a smaller r (via the software tool) for maintaining a smaller fitness difference between two optimums of a test instance, and a larger r for a larger fitness difference between the optima. As an example, we provide the information of optimal solutions (global and local) of two *SCMI* instances namely *Instance-4* and *Instance-5* in Table 3.2. The *Instance-4* is characterized by: $\mathbf{w}$={11, 21, 14, 9, 5, 12, 17, 20, 19, 3, 6, 7, 13, 10, 18}, $\mathbf{p}$={176, 336, 224, 144, 80, 192, 272, 320, 304, 48, 96, 112, 208, 160, 288}, and C=3 to 21 and *Instance-5* is characterized by: $\mathbf{w}$={19, 5, 21, 20, 18, 13, 10, 7, 11, 14, 17, 12, 9, 6, 3}, $\mathbf{p}$={399, 105, 441, 420, 378, 273, 210, 147, 231, 294, 357, 252, 189, 126, 63}, and C=3 to 21. We generate these two instances with the similar parameter settings as *Instance-3*, except the parameter r which is set to 15 for *Instance-4* and 20 for *Instance-5*. According to Table 3.2, the fitness differences between the global optima and local optima of *Instance*-4 are

Table 3.2: The fitness values of the optima of *Instance*-4 and *Instance*-5.

	Instance-4		*Instance*-5	
Optimum	Location	Fitness	Location	Fitness
1	000100000000000	144	000000000000100	189
2	000010000100000	128	010000000000001	168
3	000000000110000	144	000000000000011	189
4	000000000001000	112	000000010000000	147

16(144-128) and 32(144-112), respectively. However, the fitness differences between the global and local optima of *Instance*-5 are 21(189-168) and 42(189-147) which are larger than 16 and 32, respectively.

Distribution of Optima

It is well-known that the multimodal instances with irregular distribution of optima are difficult to solve than the regular distribution of optima. Therefore, the proposed instance generation framework is designed for the multimodal 0-1 knapsack instances with irregular distribution of optima. For example, we can calculate the hamming distances between the optima of *Instance*-2, in Table 3.1. It can be found that these optima are maintaining the different distance between them. This proves that they are irregularly distributed in the search space of *Instance*-2.

3.6 The Proposed Multimodal 0-1 Knapsack Test Instances

Table 3.3 to 3.5 show three different sets of strongly correlated multimodal 0-1 knapsack test instances. In these tables, the first column shows the names of the multimodal instances. The second column gives the generator parameters used for producing the exemplary multimodal 0-1 knapsack test instances. The third column provides the 0-1 knapsack parameters. The fourth column shows the number of dimensions (Dim) of the problems. The fifth and sixth column shows the number of optima N_{Opt} and fitness of the optima, respectively.

In Table 3.3, the first set contains four different 15-dimensional multimodal 0-1 knapsack instances namely $SCMI$-1 to $SCMI$-4. These instances have the number of optima ranging from 2 to 11, but their fitness values are different from each other.

Table 3.4 shows the second set of strongly correlated multimodal instances. This set also contains four different instances namely $SCMI$-5 to $SCMI$-8, each of which has the same number of optima with the same fitness value. However, the number of dimensions of these instances are ranging from 10 to 35.

Table 3.3: Description of the first set of multimodal 0-1 knapsack test instances.

Instance	Gen. param.	Knapsack parameters	Dim	N_{Opt}	Fitness
$SCMI$-1	n=15, k=5	**w**={10, 17, 13, 11, 18, 21, 12, 14, 3, 19, 6, 20, 9, 7, 5}, **p**={60, 102, 78, 66, 108, 126, 72, 84, 18, 114, 36, 120, 54, 42, 30}, C=9	15	2	54
$SCMI$-2	n=15, k=5	**w**={10, 17, 13, 11, 18, 21, 12, 14, 3, 19, 6, 20, 9, 7, 5}, **p**={60, 102, 78, 66, 108, 126, 72, 84, 18, 114, 36, 120, 54, 42, 30}, C=14	15	4	84
$SCMI$-3	n=15, k=5	**w**={10, 17, 13, 11, 18, 21, 12, 14, 3, 19, 6, 20, 9, 7, 5}, **p**={60, 102, 78, 66, 108, 126, 72, 84, 18, 114, 36, 120, 54, 42, 30}, C=19	15	8	114
$SCMI$-4	n=15, k=5	**w**={10, 17, 13, 11, 18, 21, 12, 14, 3, 19, 6, 20, 9, 7, 5}, **p**={60, 102, 78, 66, 108, 126, 72, 84, 18, 114, 36, 120, 54, 42, 30}, C=21	15	11	126

Table 3.4: Description of the second set of multimodal 0-1 knapsack test instances.

Instance	Gen. param.	Knapsack parameters	Dim	N_{Opt}	Fitness
$SCMI$-5	n=10, k=5	**w**={5, 7, 6, 9, 10, 12, 3, 13, 11, 14},**p**={30, 42, 36, 54, 60, 72, 18, 78, 66, 84}, C=13	10	3	78
$SCMI$-6	n=15, k=5	**w**={5, 17, 7, 20, 18, 11, 19, 14, 10, 13, 21, 9, 3, 6, 12}, **p**={30, 102, 42, 120, 108, 66, 114, 84, 60, 78, 126, 54, 18, 36, 72}, C=13	15	3	78
$SCMI$-7	n=20, k=5	**w**={22, 14, 6, 21, 5, 12, 10, 24, 11, 26, 28, 9, 19, 7, 13, 25, 20, 17, 3, 18}, **p**={132, 84, 36, 126, 30, 72, 60, 144, 66, 156, 168, 54, 114, 42, 78, 150, 120, 102, 18, 108}, C=13	20	3	78
$SCMI$-8	n=25, k=5	**w**={18, 7, 31, 35, 34, 9, 12, 25, 14, 10, 20, 3, 26, 11, 36, 13, 33, 6, 24, 5, 28, 22, 17, 21, 19}, **p**={108, 42, 186, 210, 204, 54, 72, 150, 84, 60, 120, 18, 156, 66, 216, 78, 198, 36, 144, 30, 168, 132, 102, 126, 114}, C=13	25	3	78
$SCMI$-9	n=30, k=5	**w**={33, 36, 20, 14, 3, 13, 21, 9, 18, 10, 40, 26, 38, 34, 22, 7, 31, 17, 19, 35, 12, 24, 28, 5, 6, 25, 11, 41, 42, 37}, **p**={198, 216, 120, 84, 18, 78, 126, 54, 108, 60, 240, 156, 228, 204, 132, 42, 186, 102, 114, 210, 72, 144, 168, 30, 36, 150, 66, 246, 252, 222}, C=13	30	3	78
$SCMI$-10	n=35, k=5	**w**={33, 9, 22, 12, 36, 34, 37, 19, 14, 3, 10, 20, 24, 42, 5, 49, 44, 13, 31, 6, 41, 28, 35, 47, 48, 40, 50, 7, 26, 11, 18, 17, 38, 21, 25}, **p**={198, 54, 132, 72, 216, 204, 222, 114, 84, 18, 60, 120, 144, 252, 30, 294, 264, 78, 186, 36, 246, 168, 210, 282, 288, 240, 300, 42, 156, 66, 108, 102, 228, 126, 150}, C=13	35	3	78

In Table 3.5, the third set of multimodal 0-1 knapsack instances are demonstrated. This set of instances are different from the previous two sets, as it contains the instances with the increasing number of dimensions, optima, and fitness of the optima.

In addition to the instances provided in Table 3.3 to 3.5, the user can generate more

Table 3.5: Description of the third set of multimodal 0-1 knapsack test instances.

Instance	Gen. param.	Knapsack parameters	Dim	N_{Opt}	Fitness
$SCMI$-11	n=40, k=5	**w**={38, 3, 22, 21, 35, 28, 6, 11, 56, 34, 31, 19, 33, 10, 55, 52, 37, 50, 59, 40, 20, 12, 49, 18, 61, 42, 47, 13, 5, 44, 48, 9, 26, 17, 14, 7, 41, 24, 25, 36}, **p**={228, 18, 132, 126, 210, 168, 36, 66, 336, 204, 186, 114, 198, 60, 330, 312, 222, 300, 354, 240, 120, 72, 294, 108, 366, 252, 282, 78, 30, 264, 288, 54, 156, 102, 84, 42, 246, 144, 150, 216}, C=10	40	2	60
$SCMI$-12	n=50, k=10	**w**={73, 41, 22, 25, 47, 20, 68, 34, 74, 66, 24, 44, 35, 55, 21, 18, 6, 37, 19, 69, 7, 3, 61, 26, 9, 14, 67, 38, 12, 11, 48, 62, 33, 52, 31, 13, 50, 36, 72, 49, 17, 28, 56, 10, 5, 40, 59, 70, 42, 65}, **p**={803, 451, 242, 275, 517, 220, 748, 374, 814, 726, 264, 484, 385, 605, 231, 198, 66, 407, 209, 759, 77, 33, 671, 286, 99, 154, 737, 418, 132, 121, 528, 682, 363, 572, 341, 143, 550, 396, 792, 539, 187, 308, 616, 110, 55, 440, 649, 770, 462, 715}, C=20	50	9	220
$SCMI$-13	n=60, k=15	**w**={33, 47, 80, 69, 28, 66, 73, 67, 14, 26, 84, 56, 70, 13, 6, 72, 49, 31, 44, 61, 87, 35, 5, 19, 24, 10, 65, 55, 59, 74, 21, 12, 68, 52, 18, 25, 81, 41, 82, 38, 22, 7, 3, 34, 40, 20, 88, 11, 50, 17, 9, 37, 93, 79, 36, 91, 48, 62, 42, 76}, **p**={528, 752, 1280, 1104, 448, 1056, 1168, 1072, 224, 416, 1344, 896, 1120, 208, 96, 1152, 784, 496, 704, 976, 1392, 560, 80, 304, 384, 160, 1040, 880, 944, 1184, 336, 192, 1088, 832, 288, 400, 1296, 656, 1312, 608, 352, 112, 48, 544, 640, 320, 1408, 176, 800, 272, 144, 592, 1488, 1264, 576, 1456, 768, 992, 672, 1216}, C=25	60	18	400
$SCMI$-14	n=70, k=20	**w**={104, 42, 87, 9, 97, 70, 13, 69, 5, 72, 76, 11, 82, 10, 62, 110, 34, 7, 24, 107, 35, 80, 52, 55, 36, 28, 68, 37, 3, 73, 100, 88, 79, 50, 33, 41, 12, 66, 20, 94, 31, 6, 61, 81, 44, 56, 38, 74, 48, 22, 47, 98, 103, 65, 96, 18, 93, 21, 67, 49, 14, 84, 59, 25, 91, 109, 19, 17, 40, 26}, **p**={2184, 882, 1827, 189, 2037, 1470, 273, 1449, 105, 1512, 1596, 231, 1722, 210, 1302, 2310, 714, 147, 504, 2247, 735, 1680, 1092, 1155, 756, 588, 1428, 777, 63, 1533, 2100, 1848, 1659, 1050, 693, 861, 252, 1386, 420, 1974, 651, 126, 1281, 1701, 924, 1176, 798, 1554, 1008, 462, 987, 2058, 2163, 1365, 2016, 378, 1953, 441, 1407, 1029, 294, 1764, 1239, 525, 1911, 2289, 399, 357, 840, 546}, C=31	70	38	651

challenging set of strongly correlated and subset sum multimodal 0-1 knapsack instances using the developed software framework.

3.7 Experimental Studies

To verify the usefulness of the proposed benchmark instances, experimental studies are carried out to study the effect of changing dimensionality, the number of optima, and the fitness of optima on the performance of a niching algorithm. For this purpose, we propose a binary niching method namely B-SPSO which is adopted and evaluated using the proposed benchmark instances.

3.7.1 The Proposed Binary SPSO (B-SPSO) Niching Method

A niching method is usually embedded within an evolutionary algorithm, with an aim to locate multiple optimal solutions within a single run. Some classical niching methods include *fitness sharing* [Goldberg and Richardson 1987], *clearing* [Petrowski 1996], *crowding* [Mahfoud 1995], and *speciation* [Li et al. 2002]. In addition, niching methods have also developed in conjunction with PSO or DE, e.g., speciation-based PSO (SPSO) [Li 2004], NichePSO [Brits et al. 2002], and DE/nrand [Epitropakis et al. 2012]. These niching methods have been largely designed for the continuous problems. However, little work has been done on niching methods that are specifically designed for solving discrete problems.

For this chapter, we develop a binary species-based PSO (B-SPSO) niching method by incorporating the idea of speciation [Li et al. 2002, Li 2004] into the standard binary PSO model (BPSO) [Kennedy and Eberhart 1997]. Kennedy and Eberhart [Kennedy and Eberhart 1997] developed the first BPSO to tackle the binary optimization problems. It uses a number of particles which fly around the binary search space to find the best solution. At k-th iteration, the velocity and position of the i-th particle are modified according to the following equations:

$$\mathbf{v}_i^{k+1} = \mathbf{v}_i^k + c_1 r_1^k (\mathbf{p}_i^k - \mathbf{x}_i^k) + c_2 r_2^k (\mathbf{p}_{g,i}^k - \mathbf{x}_i^k) \tag{3.4}$$

$$\mathbf{x}_i^{k+1} = \begin{cases} 0 & \text{if } rand() > S(\mathbf{v}_i^{k+1}), \\ 1 & \text{otherwise.} \end{cases} \tag{3.5}$$

where k represents a current iteration, $\mathbf{p}_i$ and $\mathbf{p}_{g,i}$ are the personal best position and local best position, respectively, $rand()$ is a random number drawn from the uniformly distributed $\mathcal{U}(0,1)$, c_1 and c_2 are cognitive and social weights, r_1^k and r_2^k are cognitive and social random variables in range [0,1], and $S(\mathbf{v}_i^k)$ denotes a sigmoid function which can be determined by the following equation:

$$S(\mathbf{v}_i^{k+1}) = \frac{1}{1 + e^{-\mathbf{v}_i^{k+1}}}. \tag{3.6}$$

In binary SPSO (B-SPSO), speciation involves the splitting of a single population into a number of sub-populations (species) according to their similarities measured by the

hamming distance (d^{Ham}) [Hamming 1950]. The smaller the hamming distance between two population individuals, the more similar they are:

$$d^{Ham}(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{d} [x_{i,k} \neq x_{j,k}], \tag{3.7}$$

where, d represents the search dimensions, $\mathbf{x}_i$=$[x_{i1}, x_{i2}, \ldots, x_{id}]$ and $\mathbf{x}_j$=$[x_{j1}, x_{j2}, \ldots, x_{jd}]$ are two binary position of the particles i and j from the BPSO population. In this equation, the statement $x_{i,k} \neq x_{j,k}$ within the square bracket will return 1 if it is true, and 0 otherwise.

The formation of species depends on a user-defined parameter r_s, which denotes the radius measured in Hamming distance from the center of a species to its boundary. The center of a species, so-called **species seed**, is always the best-fit individual in the species. All the population individuals that fall within the r_s distance from the species seed are classified as the same species. Fig. 3.3 shows an example of the species seeds and non-dominating individuals of the seven different species. After creating the species, B-SPSO determines their dominant members i.e., species seeds according to Algorithm 4 (see Chapter 2), where the distance d and niche radius r_s should be measured in terms of Hamming distance (see Eq. (3.7)). In the followings, we will describe the Pseudocode for the proposed binary SPSO (B-SPSO) niching method.

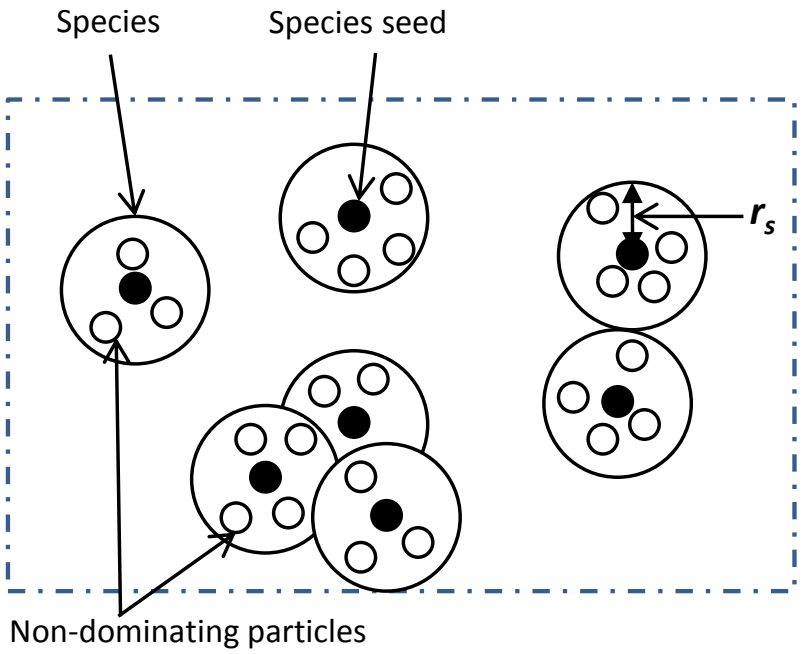


Figure 3.3: Illustration of the species seeds and non-dominating individuals of the seven species.

Algorithm 7 Pseudocode of the binary SPSO (B-SPSO) niching method.

```
1: Randomly generate an initial population
2: repeat
3:     for i=1 to popsize do
4:         if f(x_i) > f(p_i) then p_i ← x_i;
5:         end if
6:     end for
7:     L_sorted ← SORT_Desc(P);                    ▷ P is the list of particles.
8:     S ← GET_SEEDS(L_sorted);                    ▷ Algorithm 4
9:     for i=1 to popsize do
10:        f(p_{g,i}) ← f(x_{seed,i});
11:        p_{g,i} ← x_{seed,i};
12:        Eq. (3.4);
13:        Eq. (3.5);
14:    end for
15: until the termination criteria is met
```

Algorithm 7 demonstrates the pseudocode of binary SPSO (B-SPSO) niching method. In this algorithm, lines 3 to 6 are used to determine the personal best position of the particles. Line 7 sorts all the particles in descending order of their fitness values. Line 8 determines the species seeds for the current population according to Algorithm 4. Line 9 to 14 are used to update $\mathbf{p}_{g,i}$, $\mathbf{v}_i$, and $\mathbf{x}_i$, respectively. Finally, line 15 is used to check the termination criteria of B-SPSO.

3.7.2 Performance Measure

The performance of the binary SPSO method is measured using three popular performance metrics namely the peak ratio, success rate, and an average number of function evaluations [Li et al. 2013, Thomsen 2004, Li 2010, Das et al. 2011, Fieldsend 2014]. For clarity, the definitions of these performance metrics are provided below.

- *Peak Ratio (PR)*: Given a fixed number of function evaluations (*maxFEs*), the peak ratio measures the average percentage of all known optima over multiple runs:

$$PR = \frac{\sum_{k=1}^{N_R} NGF_k}{N_{gpt} \times N_R}, \tag{3.8}$$

  where NGF_k represents the number of optima found at the end of k-th run, N_{opt} indicates the number of known optima, and N_R represents the number of runs.

- *Success Rate (SR)*: The success rate measures the percentage of successful runs (a successful run is defined as a run where all known optima are found) out of all runs:

$$SR = \frac{NS_R}{N_R}, \tag{3.9}$$

  where NS_R represents the number of successful runs.

- *Average Number of Function Evaluations (AvgFEs)*: This performance metric measures the number of function evaluations required to find out all known optima. We calculate the average FEs over multiple runs:

$$AvgFEs = \frac{\sum_{k=1}^{N_R} FE_k}{N_R}, \tag{3.10}$$

where FE_k represents the number of evaluations used in the k-th run. If the test algorithm unable to locate all the optima by the *maxFEs*, then *maxFEs* is used when calculating the average *FEs*.

3.7.3 Experimental Settings

The population size (*popSize*) and the maximum number of function evaluations (*maxFEs*) are two important parameters of any niching method. Generally, these two parameters are empirically specified. For example, in SPSO [Parrott and Li 2006], *popSize*=30 and *maxFEs*=60000 were used for the optimization of the problems with 1 to 5 global optima. In [Qu et al. 2012], for locating 1 to 5 global optima, *popSize* was set to 50 and *maxFEs* was set to 10000, and for 18 global optima, *popSize* and *maxFEs* were set to 250 and 100000, respectively. In our experiments, we choose a fixed *popSize*=250 and *maxFEs*=100000, which would be sufficient for locating all the optimal solutions of the proposed multimodal 0-1 knapsack instances.

For the test instances used in this chapter, since the exact number of optima and the distances between them are known, a value slightly smaller than the distance between two closest optima can be chosen as the species radius r_s. With this, the binary SPSO should be able to sufficiently distinguish two different optima. To choose the best value for r_s, we have conducted several experiments using binary SPSO with different niche radius values. After the preliminary study, the value for the niche radius r_s is set to 0.1. In addition, the other parameters in binary SPSO were fixed to the values of w=1, $c_1 = 2.05$, $c_2 = 2.05$, and $V_{max} = 4$, as suggested in [Kennedy and Eberhart 1997, Parrott and Li 2006].

The binary SPSO will be terminated when either all the optimal solutions are found or the maximal number of function evaluations is reached. Note that we run the binary SPSO 30 times for each multimodal 0-1 knapsack test instance to remove the randomness of the algorithm.

3.7.4 Results and Discussions

In this section, the experimental results obtained by the binary SPSO are presented. The binary SPSO method is used as the basic niching methods for the multimodal 0-1 knapsack problem. The obtained results are intended as benchmarks for the multimodal 0-1 knapsack test instances.

Table 3.6: The mean and standard deviation of SR, PR, and $AvgFEs$ over 30 runs to locate the optima of the first set of multimodal test instances.

Dim	Instance	N_{Opt}	PR	SR	$AvgFEs$
15	$SCMI$-1	2	**0.45(0.01)**	**0.10(0.01)**	**93208.33(772.86)**
	$SCMI$-2	4	0.50(0.01)	0.07(0.01)	97016.67(430.00)
	$SCMI$-3	8	0.42(0.01)	0.00(0.00)	100000.00(0.00)
	$SCMI$-4	11	0.46(0.00)	0.00(0.00)	100000.00(0.00)

Table 3.7: The mean and standard deviation of SR, PR, and $AvgFEs$ over 30 runs for locating the optima of the second set of multimodal test instances.

Dim	Instance	N_{Opt}	PR	SR	$AvgFEs$
10	$SCMI$-5	3	**1.00(0.00)**	**1.00(0.00)**	**2558.33(71.14)**
15	$SCMI$-6	3	0.63(0.01)	0.27(0.01)	87241.67(933.80)
20	$SCMI$-7	3	0.02(0.00)	0.00(0.00)	100000.00(0.00)
25	$SCMI$-8	3	0.00(0.00)	0.00(0.00)	100000.00(0.00)
30	$SCMI$-9	3	0.00(0.00)	0.00(0.00)	100000.00(0.00)
35	$SCMI$-10	3	0.00(0.00)	0.00(0.00)	100000.00(0.00)

Table 3.6 and 3.7 show the results (mean $\pm$ standard deviation) of peak ratio, success rate, and an average number of function evaluations for locating the optima of each of the proposed multimodal test instances, using Eq. (3.8), (3.9), and (3.10), respectively. In these tables, the first, second, and third column represent the number of dimensions (Dim), instances name, and the number of optima (N_{Opt}) of the test instances, respectively. The fourth, fifth, and sixth column show the peak ratio (PR), success rate (SR), and the averaged number of function evaluations ($AvgFEs$), respectively. In these tables, the best results provided by the B-SPSO are marked in bold.

In Table 3.6, the peak ratio (PR) shows that the B-SPSO can locate on average 45% optima in all the runs for the first set of instances. However, the obtained success rates SR show that B-SPSO could not locate all the optima in all the runs within the given amount of function evaluations, particularly for the instances $SCMI$-3 and $SCMI$-4. The particular reason is that since the optima are irregularly distributed, some of the optima might be far away from the other optima in the search space. To locate them, better search ability is required which seems to be lacking in the binary version of SPSO.

As can be seen in Table 3.7, with respect to PR and SR, the B-SPSO can locate 100% optima of a 10-dimensional multimodal instance in 100% runs with $AvgFEs$=2558.33. However, as the number of dimensions increase, both PR and SR decrease significantly for the 15-dimensional to 35-dimensional multimodal 0-1 knapsack instances. Particularly, B-SPSO was unable to locate even a single optima for 25-dimensional to 35-dimensional multimodal 0-1 knapsack instances. The reason might be that the search spaces of the

instances $SCMI$-8 to $SCMI$-10 are much bigger than the instances $SCMI$-5 to $SCMI$-7. Hence, B-SPSO may not have explored such a bigger search space comprehensively and got stuck in the local optima. For the third set, the B-SPSO was unable to locate the multiple optima of the third set instances, and thus the results are not provided in this section.

3.8 Chapter Summary

This chapter addresses an important research gap in designing test suites for the discrete multimodal optimization problems. A new framework is proposed for generating a diverse set of challenging multimodal test instances using the classical 0-1 knapsack problem. The framework offers the flexibility to control the number of dimensions, number of optima, distribution of optima, and fitness of optima of multimodal test instances. To demonstrate the framework, we have proposed 14 strongly correlated multimodal benchmark instances for the 0-1 knapsack problem. For illustration, we have provided experimental results using the proposed binary SPSO (B-SPSO) niching method to evaluate the proposed benchmark suite. The results show that the proposed binary SPSO performs well on low-dimensional multimodal instances ($Dim \leq 10$), but often performs poorly as the number of dimensions and optima increase. This suggests that the proposed framework can generate the multimodal 0-1 knapsack instances with various levels of difficulties, which can be useful for evaluating existing niching methods for discrete optimization problems.

CHAPTER 4

Better Exploration/Exploitation Balance in a Binary PSO

4.1 Introduction

In the previous chapter, we have seen that the B-SPSO is unable to locate a good number of optima for the high-dimensional 0-1 knapsack test instances. The reason of this shortcoming is the poor search ability of B-SPSO. To overcome this issue, this chapter introduces a time-varying transfer function based binary PSO (TV_T-BPSO) for the proposed B-SPSO niching method. The proposed TV_T-BPSO has the ability to provide a better exploration and exploitation of binary PSO. In addition, it has the ability to make a good balance between exploration and exploitation. The rest of the chapter is organized as follows. Section 4.2 provides the related work of this chapter. Section 4.3 provides an analysis on the behaviour of existing transfer functions of BPSO. Section 4.4 presents the details of the proposed TV_T-BPSO, followed by Section 4.5 on a simple example to demonstrate the exploration and exploitation ability of TV_T-BPSO. Section 4.6 first presents some statistical results that compare TV_T-BPSO with four other well-known BPSO variants over 0-1 knapsack test instances. Empirical investigations on the influence of the parameter settings of TV_T-BPSO are also conducted. Finally, the summary of this chapter is presented in Section 4.7.

4.2 Related Work

Since the past decade, many researchers have developed powerful discrete optimization methods that are inspired by nature, often referred to as metaheuristics [Yang and Xin-She 2010]. Binary particle swarm optimization (BPSO) is one popular metaheuristic algorithm first proposed by Kennedy and Eberhart [Kennedy and Eberhart 1997]. This BPSO has been used for solving various types of discrete optimization problems [Pedrasa et al. 2009,

Liao et al. 2007, Jarboui et al. 2008, Naeem et al. 2012, Lin et al. 2016, Han et al. 2017]. However, it has been observed that BPSO seems to be lacking the exploration capability that is needed for obtaining high-quality solutions [Bansal and Deep 2012, Mirjalili and Lewis 2013, Liu et al. 2011]. Many BPSO variants have been developed to tackle this issue [Bansal and Deep 2012, Mirjalili and Lewis 2013, Liu et al. 2011, Ting et al. 2006, Tassopoulos and Beligiannis 2012a;b, Liu et al. 2015a, Han et al. 2017, Liu et al. 2015b, Wang et al. 2013]. However, the existing BPSO variants still have the issue of maintaining a good balance between exploration and exploitations, since too much exploration often degrades the fine-tuning ability of BPSO. On the other hand, too much exploitation gives more refining capability but it may adversely drive the search towards local optimal solutions.

In BPSO, the transfer function is considered as a key operator for controlling exploration and exploitation [Bansal and Deep 2012, Mirjalili and Lewis 2013]. Using an inappropriate transfer function can substantially degrade the performance of the BPSO. The sigmoid transfer function (S_T) typically employed in BPSO has several identified limitations [Bansal and Deep 2012, Mirjalili and Lewis 2013, Wang et al. 2008b]. Specifically, this transfer function is unable to provide BPSO a sufficient amount of diversity causing an imbalance between exploration and exploitation.

In the past few years, two different approaches have been proposed to modify the original BPSO for improving its performance. The first approach focused on designing new rules to update the particle's velocity and position for BPSO. For example, in [Jeong et al. 2010], the quantum-inspired BPSO (QBPSO) adopts a Q-bit individual for the probabilistic representation to replace the velocity update equation in BPSO. In [Modiri and Kiasaleh 2011], a new velocity update equation is proposed based on the observation of the personal best influence and initial velocity values of BPSO. Another modified BPSO namely Binary Accelerated Particle Swarm Algorithm (BAPSA) is proposed in [Beheshti et al. 2013]. In this modified BPSO, a new velocity update equation adopted which is designed based on Newtonians motion laws. Recently, Banka and Dara [Banka and Dara 2015] proposed a hamming distance based binary particle swarm optimization (HDBPSO) for features election, classification and validation. In this BPSO, the hamming distance is used as a proximity measure for updating the velocity of the particle's in binary PSO. In another improved BPSO [Han et al. 2017], a new position update rule is used to enhance the performance of original BPSO for gene selection from microarray data. However, the above-mentioned BPSOs have several shortcomings, e.g., no strategy to tune the new parameters, computationally expensive, and relatively hard to implement [Jordehi and Jasni 2015, Wang et al. 2017]. Therefore, they are not generally applicable to a wide range of discrete optimization problems.

In contrast, the second approach focused on replacing the sigmoid transfer function with new transfer functions in order to update each particle's position in a way to encour-

age a better exploration of the search space. The V-shaped and linear normalized transfer function is one such scheme proposed in [Bansal and Deep 2012, Mirjalili and Lewis 2013, Liu et al. 2011]. It is reported that the V-shaped transfer function based BPSOs [Mirjalili and Lewis 2013, Liu et al. 2011] and linear normalized transfer function based BPSO [Bansal and Deep 2012] outperform the original BPSO [Kennedy and Eberhart 1997] and some other well-known BPSOs [Lee et al. 2008, Shen et al. 2004, Wang et al. 2008a] over a set of low-dimensional optimization problems [Bansal and Deep 2012, Suganthan et al. 2005]. The reason is also reported in [Bansal and Deep 2012, Mirjalili and Lewis 2013, Liu et al. 2011] that these transfer functions have the ability to promote more exploration compared to the conventional sigmoid transfer function.

To examine the general exploration and exploitation capability of the BPSO, it is necessary to analyze the performance of the existing transfer functions [Bansal and Deep 2012, Mirjalili and Lewis 2013, Liu et al. 2011] over a diverse set of both low-dimensional and high-dimensional discrete test problems to see whether they can maintain a good balance between exploration and exploitation.

4.3 Search Behaviour Analysis of BPSO

For the clarity, this section first revisits the BPSO algorithm, and then examines the behaviour of BPSO employing existing transfer functions to identify their merits and shortcomings in balancing between exploration and exploitation.

4.3.1 Binary Particle Swarm Optimization Revisited

Kennedy and Eberhart [Kennedy and Eberhart 1997] developed the first binary PSO to tackle the binary optimization problems. For clarity, we call the original BPSO as S_T-BPSO in this chapter, to differentiate it from other BPSO variants. The S_T-BPSO primarily extended the basic concept of the original PSO by using the sigmoid transfer function to transform the value of velocity from the continuous space into binary space. In this BPSO, at each iteration, the velocity v_{id} is modified according to the following equation:

$$v_{id}^{k+1} = v_{id}^{k} + c_1 r_{1d}^{k}(p_{id}^{k} - x_{id}^{k}) + c_2 r_{2d}^{k}(g_{d}^{k} - x_{id}^{k}), \tag{4.1}$$

where $(k+1)$ represents the current iteration, d represents the dimension, r_{1d} and r_{2d} are two random numbers drawn from the uniformly distribution $\mathcal{U}(0,1)$, and $c_1 > 0$ and $c_2 > 0$ are the cognitive and social weights, respectively.

After updating the velocity, the position x_{id} is modified according to the following:

$$x_{id}^{k+1} = \begin{cases} 0 & \text{if } rand() \geq S_T(v_{id}^{k+1}), \\ 1 & \text{otherwise,} \end{cases} \tag{4.2}$$

where $S_T(v_{id}^{k+1})$ is a sigmoid transfer function which denotes the probability (in the range of [0,1]) for a bit that takes the value 0 or 1:

$$S_T(v_{id}^{k+1}) = \frac{1}{1 + e^{-v_{id}^{k+1}}}. \tag{4.3}$$

It is reported that BPSO has the difficulties to provide a sufficient amount of exploration, because of the conventional sigmoid transfer function [Ting et al. 2006, Wang et al. 2008b, Lee et al. 2008, Shen et al. 2004, Yang et al. 2014]. To overcome this, the modified BPSO with a linear normalized and two different V-Shaped transfer functions were proposed in literature [Bansal and Deep 2012, Mirjalili and Lewis 2013, Liu et al. 2011]. In the followings, we will provide a brief analysis of the behavior of sigmoid function, linear normalized function, and two different V-shaped functions to demonstrate that they have the difficulties to maintain a good balance between exploration and exploitation. However, before providing this analysis, it is important to describe the basics of two key phases of a typical BPSO search process, i.e., the exploration phase and exploitation phase.

4.3.2 Exploration and Exploitation Phase of BPSO

In BPSO, the search space is considered as a hypercube, in which a particle moves from one node to another by flipping one or more bits of its position vector $\mathbf{x}_i = (x_{i1}, \ldots, x_{id})$. BPSO uses a transfer function to determine the probability of the value of each bit x_{id} (0 or 1), which depends on the corresponding velocity. If the absolute velocity is high, then the corresponding bit will have a very high probability to be 1 (positive velocity) or 0 (negative velocity). On the other hand, if the velocity is close to zero, then the value of the corresponding bit becomes uncertain. In other words, in BPSO, a large absolute velocity leads to exploitation, and a small absolute (close to zero) velocity leads to exploration.

The exploration and exploitation phase can be determined for the binary search space by measuring the Hamming distance [Hamming 1950] between the previous position $\mathbf{x}_i^k$ and the current position $\mathbf{x}_i^{k+1}$ of the i-th particle of BPSO. The equation for measuring the Hamming distance is demonstrated in Eq. (3.7). According to the Eq. (3.7), if the measured hamming distance is found to be sufficiently large, then it can be said that BPSO is exploring the search space, otherwise, it is exploiting the search space.

4.3.3 Analysis of the Behaviour of Existing Transfer Functions

In literature, three different transfer functions can be found for BPSO, namely the *sigmoid transfer function* [Kennedy and Eberhart 1997], *linear normalized transfer function* [Bansal and Deep 2012], and *V-shaped transfer function* [Liu et al. 2011, Mirjalili and Lewis 2013]. The behaviour of these transfer functions on the performance of BPSO are examined in the following sections.

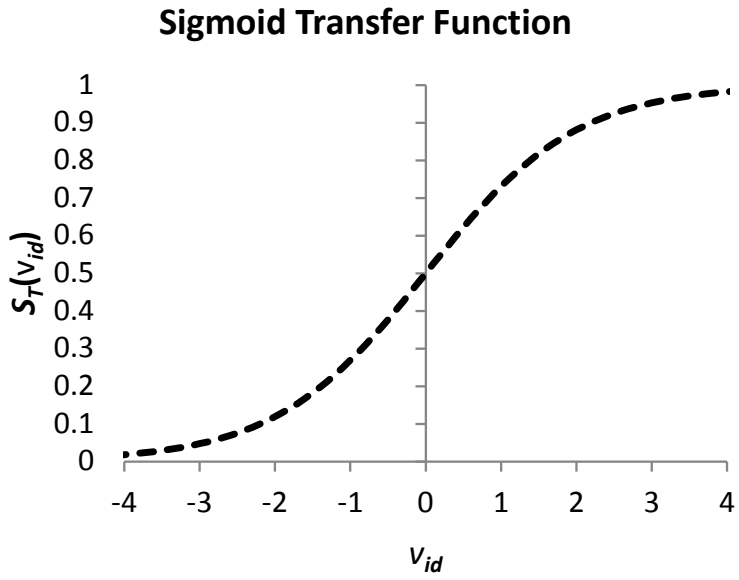


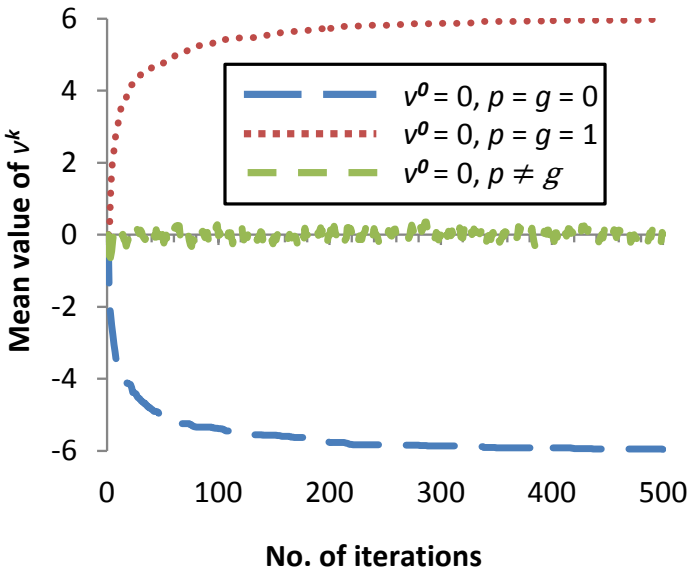
Figure 4.1: Illustration of the sigmoid transfer function (S_T).

Sigmoid Transfer Function

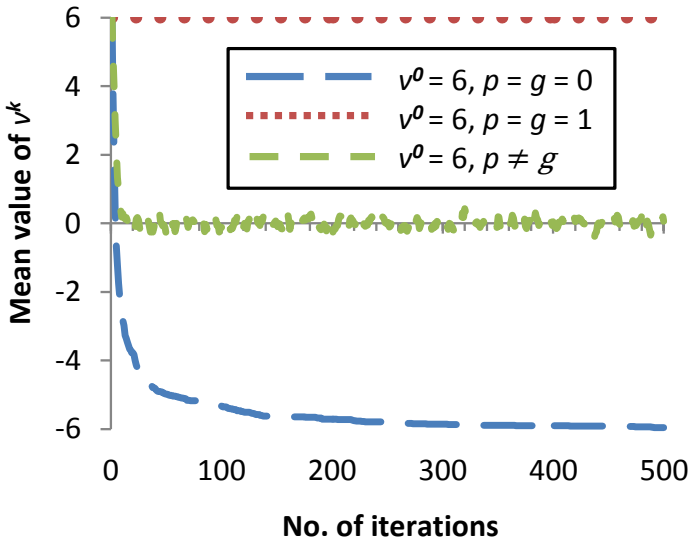
In [Kennedy and Eberhart 1997], the sigmoid transfer function (see Fig. 4.1) is introduced to map a real-valued velocity $v_{id} \in \mathbf{v}_i$ to a probability value in the range of [0,1] for changing a binary position $x_{id} \in \mathbf{x}_i$. According to Fig. 4.1, the sigmoid transfer function provides a higher bit flipping probability with the smaller absolute value of velocity and a lower bit flipping probability with a higher value of velocity. According to this, a small absolute value of velocity (close to zero) is preferable to better explore the search space in the early stages of the run. On the other hand, a large absolute value of velocity is preferable to better exploit the search space in the final stages of the run.

According to Eq. (4.1), the velocity v_{id}^{k+1} takes a smaller value when $p_{id}^k \neq g_d^k$, and takes a larger value when $p_{id}^k = g_d^k$. Since both p_{id} and g_d controls the velocity of i-th particle, S_T-BPSO cannot always maintain a smaller velocity in the early stages of the run and a higher velocity in the final stages of the run. Consequently, the sigmoid transfer function has the difficulties in maintaining a good balance between exploration and exploitation for S_T-BPSO.

To validate the above analysis, we conduct an experiment to predict the sequences of the mean value of velocity v_{id}^{k+1} of S_T-BPSO over 30 independent runs (500 iterations per run) under different p_{id}^k and g_d^k values which are unchanged over time. For the sake of simplicity, this experiment has been conducted using a basic BPSO model where a single particle and a single dimension is considered. For this experiment, the velocity is updated using Eq. (4.1) where i has been set to 1 for the single particle, d has been set to 1 for the single dimension. The other parameters have been set to their standard value as $w = 1$,



(a)



(b)

Figure 4.2: The curves of the mean value of velocity v^{k+1} of S_T-BPSO over 30 independent runs, with w=1, $c_1 = c_2 = 2$, and $v_{max} = 6$, when (a) v^0=0 and (b) v^0=v_{max}=6

$c_1 = c_2 = 2$ and $v_{max} = 6$. The position of i-th particle is updated using Eq. (4.2).

With the above settings, we can demonstrate the sequences of the mean value of velocity in two extreme scenarios: 1) when the velocity takes a small initial value v^0=0; and 2) when it takes a large initial value v^0=v_{max}=6, as shown in Fig. 4.2. By considering the curves in this figure, if we assume that the search process is in the initial stages of

the run, S_T-BPSO is expected to require a stronger exploration, i.e., a higher bit flipping probability to explore the search space. In this case, if $p \neq g$, then S_T-BPSO provides a small velocity value, from which the sigmoid transfer function can produce a stronger exploration for BPSO. However, if $p = g$, then the sigmoid transfer function cannot provide a stronger exploration due to the higher value of the velocity, as demonstrated in Fig. 4.2. As a result, most of the promising regions of the search space will remain to be unexplored; there is a higher possibility that S_T-BPSO may get trapped in local optima.

Now, if we assume that the search process is in the final stage of the run; S_T-BPSO would need a stronger exploitation, i.e., a lower bit flipping probability to improve the quality of the found solutions. According to Fig. 4.2, if $p = g$, then the sigmoid transfer function will provide a low bit flipping probability (because $v^{k+1} = 6$ and $S_T(v^{k+1}) \simeq 0$) for exploiting the search space. However, if $p \neq g$, S_T will provide a high bit flipping probability (because $v^{k+1} \simeq 0$ and $S_T(v^{k+1}) \simeq 0.5$) instead of a low bit flipping probability for S_T-BPSO. In this case, there is a higher chance that S_T-BPSO would lose the good solutions which have been found in the exploration phase.

Given the above discussion, it can be said that S_T-BPSO cannot maintain a good balance between exploration and exploitation due to the limitation of the sigmoid transfer function.

Linear Normalized Transfer Function

Bansal, *et al.* [Bansal and Deep 2012] proposed a modified BPSO termed as L_T-BPSO where a linear normalized transfer function (L_T) is used to improve the exploration ability of S_T-BPSO. The linear normalize transfer function of L_T-BPSO is illustrated in Fig. 4.3 and mathematically defined by the following:

$$L_T(x_{id}^k, v_{id}^{k+1}) = \frac{x_{id}^k + v_{id}^{k+1} + v_{max}}{1 + 2v_{max}}. \tag{4.4}$$

In L_T-BPSO, the velocity of i-th particle is modified according to Eq. (4.1). This velocity is then transformed into probability using Eq. (4.4), of which the position of i-th particle is updated as follows:

$$x_{id}^{k+1} = \begin{cases} 0 & \text{if } rand() \geq L_T(x_{id}^k, v_{id}^{k+1}), \\ 1 & \text{otherwise.} \end{cases} \tag{4.5}$$

It can be observed that L_T-BPSO and S_T-BPSO uses the same velocity and position updating rules, except the transfer function for updating the particle position. Therefore, L_T-BPSO should experience the same difficulties as S_T-BPSO as described earlier. However, the only difference between these BPSOs is that the linear normalized transfer function can provide a better exploration compared to the sigmoid transfer function. For example, if v_{id}=-3 and x_{id}=0, then according to the Eq. (4.3) and Fig. 4.4, there is a

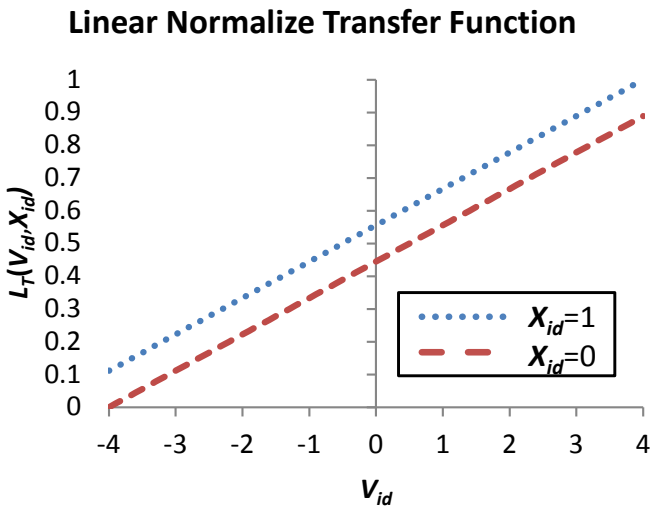


Figure 4.3: Illustration of the linear normalized transfer function (L_T).

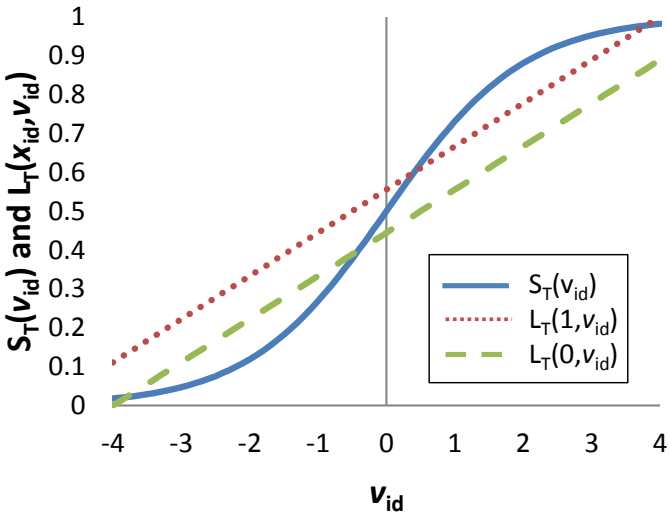


Figure 4.4: Comparison between the sigmoid and linear normalized transfer function.

probability of 0.047 that x_{id} will be flipped from 0 to 1. Now according to the Eq. (4.4), the probability of flipping x_{id} from 0 to 1 is 0.111 which is greater than 0.047. With the higher bit flipping probability, there is a higher chance that the bits of x_{id} will be flipped in the next iteration.

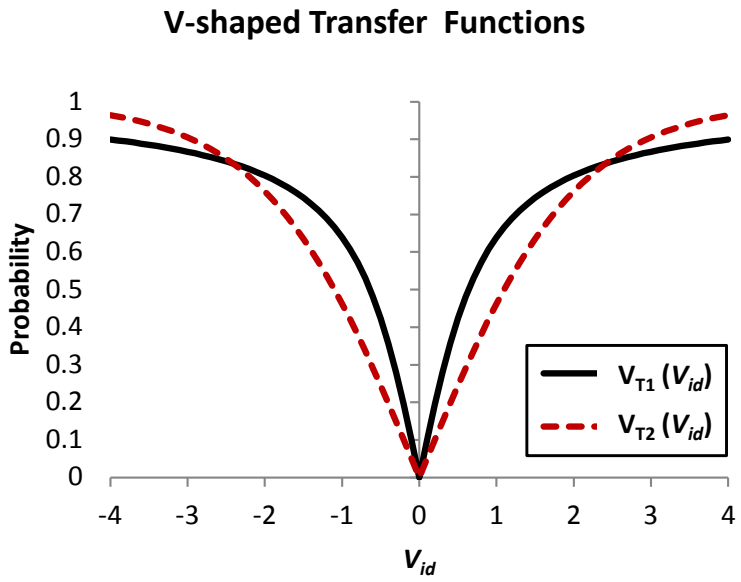


Figure 4.5: Illustration of two different V-shaped transfer functions V_{T1} and V_{T2}.

V-shaped Transfer Functions

Two different V-shaped transfer functions were proposed in literature [Liu et al. 2011, Mirjalili and Lewis 2013]. The first V-shaped transfer function is termed as V_{T1} [Liu et al. 2011] and the second one V_{T2} [Mirjalili and Lewis 2013], as shown in Fig. 4.5, and the corresponding modified BPSOs are V_{T1}-BPSO and V_{T2}-BPSO.

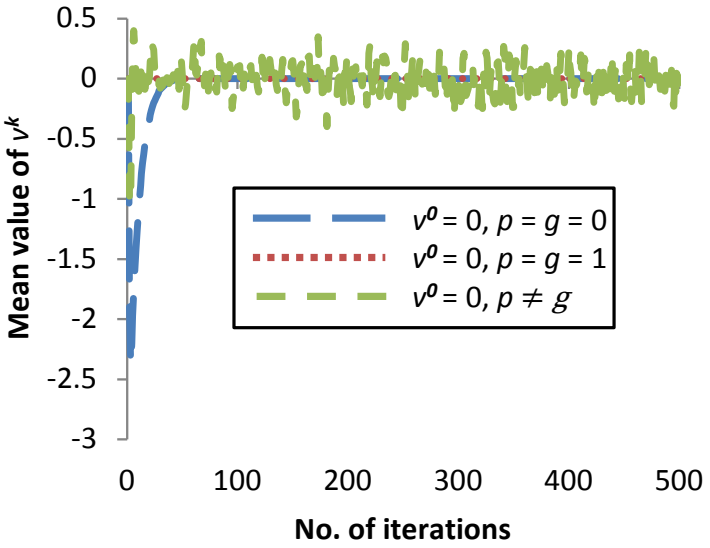
Below we provide some explanation on why these two V-shaped transfer functions are unable to produce a good balance between exploration and exploitation.

1) V-shape transfer function V_{T1}: The first V-shaped transfer function V_{T1} (as proposed for V_{T1}-BPSO) transforms a particle's velocity to a binary position using the following equation:

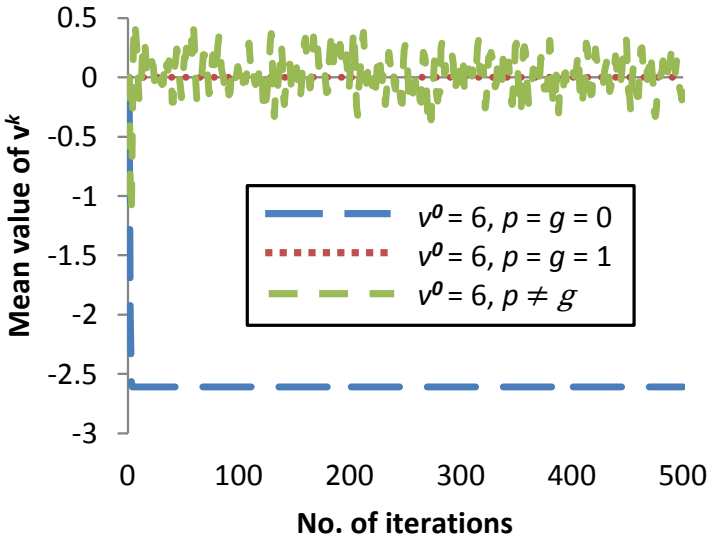
$$V_{T1}(v_{id}^{k+1}) = \begin{cases} 1 - \frac{2}{1+e^{-v_{id}^{k+1}}} & \text{if } v_{id}^{k+1} \leq 0, \\ \frac{2}{1+e^{-v_{id}^{k+1}}} - 1 & \text{otherwise.} \end{cases} \tag{4.6}$$

In V_{T1}-BPSO, Eq. (4.1) is first used to update the velocity of the i-th particle of the swarm. Eq. (4.6) is then used to transform the velocity into a probability value, by which the position of i-th particle is updated using the following:

$$x_{id}^{k+1} = \begin{cases} 0 & \text{if } rand() \leq V_{T1}(v_{id}^{k+1}) \text{ and } v_{id}^{k+1} \leq 0, \\ 1 & \text{if } rand() \leq V_{T1}(v_{id}^{k+1}) \text{ and } v_{id}^{k+1} > 0, \\ x_{id}^{k} & \text{if } rand() > V_{T1}(v_{id}^{k+1}). \end{cases} \tag{4.7}$$



(a)



(b)

Figure 4.6: The curves of the mean value of velocity v^k of V_{T1}-BPSO over 30 independent runs under different p and g values with w_{min}=0.4, w_{max}=0.9, and c_1=c_2=2, when (a) v^0=0 and (b) v^0=6, respectively.

Like S_T-BPSO and L_T-BPSO, V_{T1}-BPSO is also unable to make a good balance between exploration and exploitation either. This can be seen by the curves of the mean velocity values in Fig. 4.6. Here we ran these experiments using the similar parameter settings as in Fig. 4.2, except that the inertia weight w was linearly decreased from 0.9

to 0.4, in order to obtain the best performance for V_{T1}-BPSO.

Fig. 4.6 shows two different situations of the mean velocity value v^k of V_{T1}-BPSO. First, if $p \neq g$, then v^k fluctuates in between -0.35 and +0.35. With these velocity values, V_{T1} can produce a 16% bit flipping probability which is insufficient to explore the search space in the early stages of the run. In this situation, V_{T1}-BPSO can easily get stuck in a local optimum due to the lack of exploration. Second, if $p = g$, then v^k takes the value either 0 or -2.5. With these velocity values, V_{T1}-BPSO forces x^k to take the value 0, because v^k satisfies the first condition of Eq. (4.7). This situation also prevents V_{T1}-BPSO from exploring the search space, and the probability of getting trapped in the local optima is increased.

2) V-shape transfer function V_{T2}: This transfer function is used in V_{T2}-BPSO. Here, a tangent hyperbolic function is employed to transform a real-valued velocity into a probability value:

$$V_{T2}(v_{id}^{k+1}) = \left| \frac{2}{\pi} \arctan(\frac{\pi}{2} v_{id}^{k+1}) \right|. \tag{4.8}$$

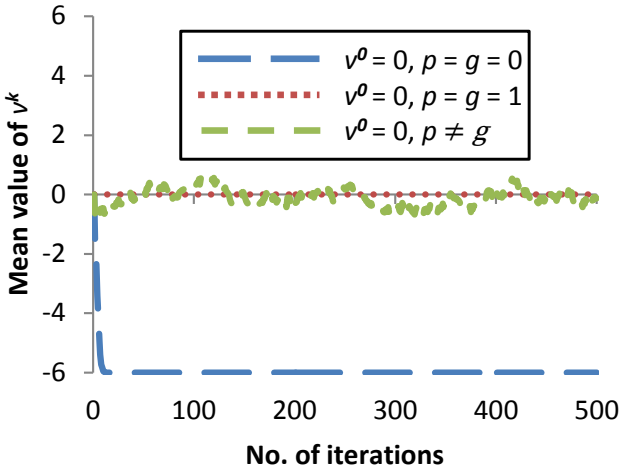
Like V_{T1}-BPSO, V_{T2}-BPSO modifies the velocity of i-th particle according to Eq. (4.1). Once the velocity is updated, Eq. (4.8) is used to transform this velocity into probability, which in turn updates the i-th particle's position according to the following:

$$x_{id}^{k+1} = \begin{cases} (x_{id}^k)^{-1} & \text{if } rand() < V_{T2}(v_{id}^{k+1}), \\ x_{id}^k & \text{otherwise.} \end{cases} \tag{4.9}$$

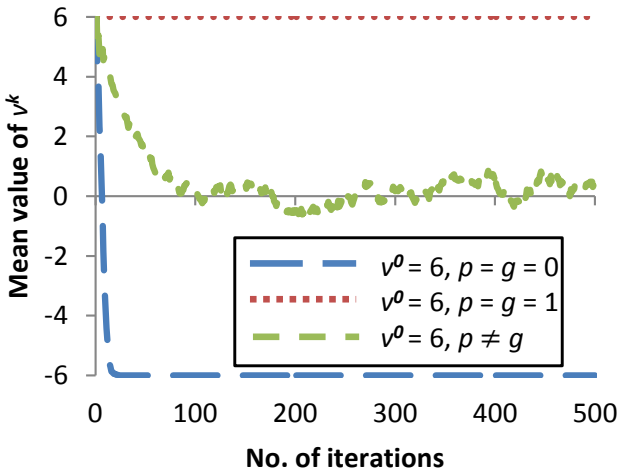
To understand the search behaviour of V_{T2}-BPSO, we conduct another experiment to show the curves of the mean velocity value v^k in Fig. 4.7 (similar to Fig. 4.5). Fig. 4.7(a) shows that if p=g=1 and $p \neq g$, then v^k takes the value of either 0 or close to 0. With these values, V_{T2} produces a 0% bit flipping probability for this modified BPSO, and thus in the next iteration, x^k will not change its position. It is clear that V_{T2}-BPSO cannot provide the exploration ability as needed in the early stages of the run.

Fig. 4.7(b) also shows that v^k reaches to the maximum limit when p=g. In this case, V_{T2} produces the highest bit flipping probability according to the Eq. (4.8). As we know that a higher bit flipping probability provides a stronger exploration which is good for V_{T2}-BPSO to explore the search space in the early stages of the run. However, this higher bit flipping probability is not desirable for V_{T2}-BPSO, since it prevents the exploitation of the search space in the final stages of the run.

Considering the above discussion in this section, clearly there is a mismatch between the velocity update equation and transfer function, resulting in the above mentioned BPSOs being unable to maintain a good balance between exploration and exploitation. To overcome this problem, this chapter proposes a new transfer function employing a



(a)



(b)

Figure 4.7: The curves of the mean velocity v^k of V_{T2}-BPSO over 30 independent runs under different p and g values with w_{min}=0.4, w_{max}=0.9, and c_1=c_2=2, when (a) v^0=0 and (b) v^0=6, respectively.

time-varying scheme in order to provide a better balance between the exploration and exploitation during the run of a BPSO.

4.4 Time-varying Transfer Function Based BPSO (TV_T-BPSO)

In this section, we first describe the design considerations of time-varying transfer function, followed by the details of the proposed TV_T-BPSO.

4.4.1 Design Considerations for the Time-varying Transfer Function

Generally speaking, in the early stages of the run, an optimization algorithm is expected to focus more on exploration to avoid being trapped in local optima, but in the later stages of the run, the algorithm needs to switch to emphasizing more on exploitation to refine the solution quality. Following this intuition, we design a dynamic transfer function for the proposed TV_T-BPSO with the following considerations:

- In the early stages of the run, the transfer function should provide a high probability of flipping all the bits of x_{id} at any value of the velocity v_{id} so that the BPSO can provide a stronger exploration.
- In the intermediate stages of the run, the BPSO should start shifting from exploration to exploitation. This can be achieved by using a transfer function able to decrease the probability of flipping all the bits of x_{id} at any value of velocity v_{id} over iterations.
- In the final stages of the run, the transfer function should provide a low probability of flipping all the bits of x_{id} at any value of velocity v_{id} so that the BPSO can provide a stronger exploitation capability.

We apply the above concepts by adopting a new control parameter φ in the sigmoid transfer function as given in Eq. (4.3). This new transfer function is given as the following:

$$TV_T(v_{id}^{k+1}, \varphi) = \frac{1}{1 + e^{-\frac{v_{id}^{k+1}}{\varphi}}}, \tag{4.10}$$

where v_{id}^{k+1} is the velocity of the i-th particle at $(k+1)$-th iteration.

Eq.(4.10) can be used as a transfer function for the proposed TV_T-BPSO. However, instead of using a fixed value for the control parameter φ, we start with a larger value for φ and gradually decrease it as the run progresses, in order to shift smoothly from more exploration to exploitation over time. This is achieved by the following:

$$\varphi = \varphi_{max} - Itr_{k+1} * \left(\frac{\varphi_{max} - \varphi_{min}}{Itr_{max}} \right), \tag{4.11}$$

where φ_{max} and φ_{min} are the bounds on the control parameter φ, Itr_{max} is the maximum number of iterations, and Itr_{k+1} is the current iteration, where $k = 0, 1, 2, ..., Itr_{max} - 1$.

We call the proposed transfer function as the time-varying transfer function (TV_T) because the shape of this transfer function changes over time depending on the different value of φ, as illustrated in Fig. 4.8. The curve $TV_T(v_{id}, \varphi_{max})$ represents the initial shape of the proposed transfer function obtained by setting φ=4. Similarly, the curve $TV_T(v_{id}, \varphi_{min})$ represents the final shape of the proposed transfer function obtained by setting φ=0.05. Other curves are obtained by the linearly decreasing values of φ.

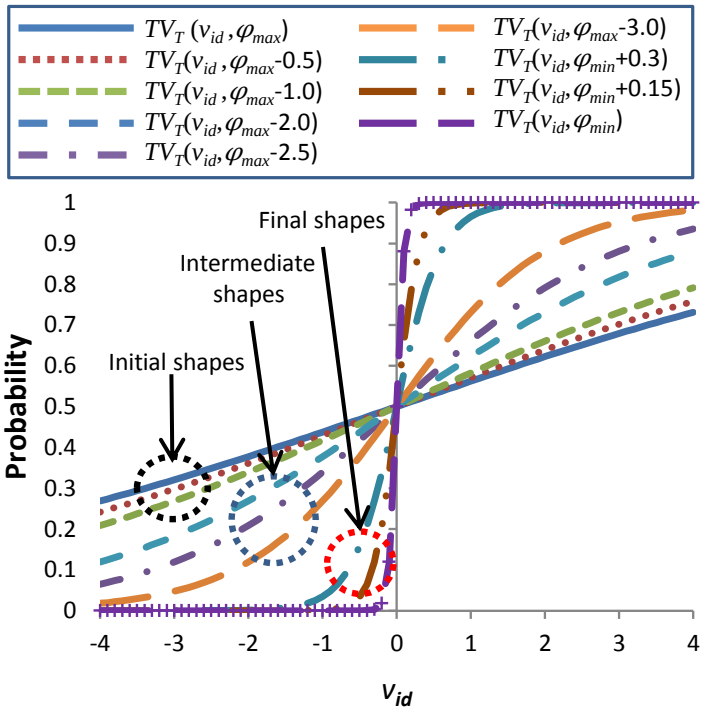


Figure 4.8: An illustration of different shapes of the time-varying transfer function (Eq. (4.10)) with different values of the control parameter φ.

As can be seen in Fig. 4.8, the curve $TV_T(v_{id}, \varphi_{max})$ linearly increases as the velocity value increases, which is much slower than the other curves. It should be noted that the probability of flipping the bits of position x_{id} increases as the slope of the curve becomes steeper. For each given velocity value, the curve $TV_T(v_{id}, \varphi_{max})$ provides the highest amount of bit flipping probability, because the curve is the closest to the probability value of 0.5 than any other curves. On the other hand, $TV_T(v_{id}, \varphi_{min})$ provides the lowest amount of bit flipping probability for changing the position x_{id} of i-th particle. Based on this observation, we propose TV_T-BPSO to adopt curves $TV_T(v_{id}, \varphi_{max})$ to $TV_T(v_{id}, \varphi_{max}$-1.0) at the start of a run in order to provide a stronger exploration; $TV_T(v_{id}, \varphi_{max}$-2.0) to $TV_T(v_{id}, \varphi_{max}$-3.0) in the intermediate stage of the run to provide a moderate level of exploration; and towards the final stage of the run $TV_T(v_{id}, \varphi_{min}$+0.3) to $TV_T(v_{id}, \varphi_{min})$ to have a stronger exploitation.

4.4.2 The Proposed TV_T-BPSO

Like S_T-BPSO, TV_T-BPSO uses Eq. (4.1) for velocity update of each particle. TV_T-BPSO uses the new time-varying transfer function Eq. (4.10) instead of the sigmoid

transfer function in Eq. (4.3) for updating the position of i-th particle:

$$x_{id}^{k+1} = \begin{cases} 1 & \text{if } rand() < TV_T(v_{id}^{k+1}, \varphi), \\ 0 & \text{otherwise.} \end{cases} \tag{4.12}$$

The pseudo-code of the proposed TV_T-BPSO is provided in Alg. 8.

4.5 Behaviour Analysis of TV_T-BPSO

This section provides an example to illustrate the differences between TV_T-BPSO and two well-known BPSOs, by considering their exploration and exploitation behaviours using a decision variable vector of 4-binary bits.

4.5.1 Exploration

Let us consider an example such that at k-th iteration of the early stages of run, the i-th particle $\mathbf{x}_i^k$ and $\mathbf{p}_i^k$ with 4-binary bits is (0010), the current $\mathbf{g}^k$ is (0011), and at the $(k+1)$-th iteration, the velocity corresponding to $\mathbf{x}_i^k$ is $\mathbf{v}_i^{k+1}$={-3.5, -3.8, 3.2, -0.1}. Note

Algorithm 8 Pseudocode of the proposed TV_T-BPSO algorithm.

Require: v_{max} ▷ the upper bound of velocity
Require: $iter_{max}$ ▷ the maximum number of iterations
Require: $popSize$ ▷ the population size

```
//initialization
iter = 0;
//randomly initialize i-th particle velocity, position, and personal best
for i=1 to popSize do
    x_i ∈ {-v_max, v_max}
    x_i ∈ {0, 1};
    p_i ← x_i;
end for
//main loop
repeat
    for i=1 to popSize do
        evaluate fitness f(x_i);
        if f(x_i) > f(p_i) then
            p_i ← x_i;
        end if
        if f(x_i) > f(g) then
            g ← x_i; f(g) ← f(x_i);
        end if
        update v_i using Eq. (4.1);
        calculate φ using Eq. (4.11);
        calculate TV_T(v_i, φ) using Eq. (4.10);
        update x_i using Eq. (4.12);
    end for
    iter = iter+1;
until iter < iter_max;
```

that at the $(k+1)$-th iteration, if p_{id}^k=g_d^k, then BPSO provides a higher value for v_{id}^{k+1}, otherwise, it provides a lower value for v_{id}^{k+1} (according to Fig. 4.2). For example, three higher values could have been chosen for the first, second, and third bit of $\mathbf{x}_i^k$ and one lower value for the fourth bit, giving us $\mathbf{v}_i^{k+1}$={-3.5, -3.8, 3.2, -0.1}. Note that the maximum velocity bound v_{max}=4 is considered here for BPSO. Based on this $\mathbf{v}_i^{k+1}$, we can compute the new position $\mathbf{x}_i^{k+1}$ using Eq. (4.3) and Eq. (4.2) for S_T-BPSO and using Eq. (4.10) and Eq. (4.12) for TV_T-BPSO, respectively.

For S_T-BPSO, we can compute the probabilities of flipping the first, second, third, and fourth bit of $\mathbf{x}_i^k$ using Eq. (4.3), which are 0.029, 0.021, 0.039, and 0.475, respectively. It is obvious that with these probabilities, only the fourth bit of $\mathbf{x}_i^k$ is highly likely to be flipped in the $(k+1)$-th iteration of the run, resulting in S_T-BPSO updating $\mathbf{x}_i^k$ from (0010) to (0011) using Eq. (4.2).

For TV_T-BPSO, let us assume that the curve $TV_T(v_{id}, \varphi_{max}\text{-}0.5)$ of Fig. 4.8 is used for determining the bit flipping probabilities of $\mathbf{x}_i^k$. From this curve, we can compute the bit flipping probabilities of the first, second, third, and fourth bit of $\mathbf{x}_i^k$, which are approximately 0.331, 0.318, 0.345 and 0.495, respectively. It can be noted that with the same $\mathbf{v}_i^{k+1}$, the bit flipping probabilities obtained by TV_T-BPSO is much higher than the bit flipping probabilities obtained by S_T-BPSO. In this case, there is a much higher chance that all of the bits of $\mathbf{x}_i^k$ of TV_T-BPSO will be flipped at $(k+1)$-th iteration, leading to the value of $\mathbf{x}_i^k$ being updated from (0010) to (1101).

The hamming distances between $\mathbf{x}_i^k$ and $\mathbf{x}_i^{k+1}$ are 1 and 4 respectively for the above two BPSOs according to Eq. (3.7). In this case, TV_T-BPSO resulting in a larger hamming distance suggesting that it has a stronger exploration capability than S_T-BPSO which only results in a smaller hamming distance.

4.5.2 Exploitation

Let us consider another example such that at the k-th iteration of the final stages of run, the i-th particle $\mathbf{x}_i^k$ and $\mathbf{p}_i^k$ with 4-binary bits is (0101), the current $\mathbf{g}^k$ is (0111), and at the $(k+1)$-th iteration, the velocity $\mathbf{v}_i^{k+1}$={-4.0, 4.0, -0.5, 4.0} has been chosen the same way as in the previous section. We can compute x_{id}^{k+1} using Eq. (4.8) and Eq. (4.9) for V_{T2}-BPSO and using Eq. (4.10) and Eq. (4.12) for TV_T-BPSO, respectively.

In case of V_{T2}-BPSO, we can compute the probability of flipping the first, second, third and, fourth bit of $\mathbf{x}_i^k$ using Eq. (4.8), which are 0.899, 0.899, 0.359, and 0.899. With these higher probabilities, there is a higher chance that the first, second, and fourth bit of $\mathbf{x}_i^k$ be flipped at $(k+1)$-th iteration, resulting in V_{T2}-BPSO updating $\mathbf{x}_i^k$ from (0101) to (1000) using Eq. (4.9).

In case of TV_T-BPSO, let us assume that the curve $TV_T(v_{id}, \varphi_{min})$ of Fig. 4.8 is used to determine the bit flipping probabilities of $\mathbf{x}_i^k$. From this curve, we can compute the bit flipping probability of four different bits of $\mathbf{x}_i^k$, which in this case are all zeros. With the

zero probabilities, $\mathbf{x}_i$ will remain unchanged at the $(k+1)$-th iteration, resulting in $\mathbf{x}_i^{k+1}$ being (0101).

The hamming distances between $\mathbf{x}_i^k$ and $\mathbf{x}_i^{k+1}$ are 3 and 0 respectively for the above two BPSOs according to Eq. (3.7). In this case, TV_T-BPSO resulting in a smaller hamming distance suggests that it has a stronger exploitation capability than V_{T2}-BPSO.

4.6 Experimental Studies

In this section, we first carry out experiments on the deterministic and non-deterministic 0-1 knapsack benchmark instances, which are commonly used benchmark problems for testing binary optimization algorithms [Bansal and Deep 2012, Zou et al. 2011a, Wang et al. 2013]. The purpose of this experiment is to identify the best values for the three parameters of TV_T-BPSO, namely φ, m, and the velocity bound v_{max}. Our second experiment is used to compare TV_T-BPSO with S_T-BPSO [Kennedy and Eberhart 1997], L_T-BPSO [Bansal and Deep 2012], V_{T1}-BPSO [Liu et al. 2011], and V_{T2}-BPSO [Mirjalili and Lewis 2013] over the low-dimensional ($4 \leq \mathrm{D} \leq 500$) 0-1 knapsack instances. The third experiment is carried out to further investigate the performance of these five BPSOs over the high-dimensional ($1000 \leq \mathrm{D} \leq 5000$) 0-1 knapsack instances.

4.6.1 Selecting the Best Values for m, v_{max}, and φ of TV_T-BPSO

In this section, an experiment is conducted to find the suitable values for the three parameters m, v_{max}, and φ. Three representative deterministic 0-1 knapsack instances Ks_8a, Ks_16b, and Ks_24d have been selected from [Bansal and Deep 2012]. And three non-deterministic 0-1 knapsack instances WCI_{100}, UCI_{500}, and $ISCI_{1000}$ have been selected, from those generated using the procedure described in Section 3.3 where the value for R is set to 1000, S is set to 0.5, and the value for n is set to 100, 500, and 1000, respectively. This experiment has been conducted with the values of m=20 to 50, v_{max}=1 to 16, φ=1 to 5, and c_1=c_2=2. The maximum number of iterations is set to 1000 for the subsequent experiments, which is sufficient to reach competitive results for the low-dimensional as well as the high-dimensional 0-1 knapsack instances of this chapter.

Table 4.1 shows the results on TV_T-BPSO, which are means over 30 independent runs. From this table, the best combinations of m, v_{max}, and φ have been presented in Table 4.2.

Table 4.2 shows that TV_T-BPSO obtained the best results when the swarm size m has been set to 30, 40, and 50. For this chapter, we choose the swarm size 40 (which is also a standard swarm size for BPSO [Wang et al. 2008b, Lee et al. 2008]) for all the following experiments.

Table 4.2 also shows that TV_T-BPSO uses the different v_{max} for obtaining the best results of six representative 0-1 knapsack instances. It can be observed that TV_T-BPSO re-

Table 4.1: Results obtained by TV_T-BPSO on Ks_8a, Ks_16b, Ks_24d, WCI_{100}, UCI_{500}, $ISCI_{1000}$ over 30 independent runs.

Instance	D	m	φ	v_{max}=2	v_{max}=4	v_{max}=6	v_{max}=8
Ks_8a	8	40	4	**3924400.00**	3923992.00	3923584.00	3920320.00
			5	3924400.00	3924400.00	3923584.00	3920728.00
		50	4	3924400.00	3923992.00	3921952.00	3919912.00
			5	3924400.00	3924400.00	3923584.00	3921544.00
Ks_16b	16	40	4	**9352998.00**	9352822.60	9341453.27	9299547.90
			5	9352647.20	9352822.60	9351945.60	9339999.47
		50	4	9352998.00	9352647.20	9338888.57	9327993.80
			5	9352822.60	9352647.20	9344899.37	9341450.33
Ks_24d	24	40	4	11782041.70	**11810362.67**	11796323.70	11777536.27
			5	11768199.07	11806689.43	11798464.23	11788310.37
		50	4	11783470.80	11803592.73	11800329.20	11782663.23
			5	11779794.33	11806170.67	11801253.70	11790710.47
Instance	D	m	φ	v_{max}=10	v_{max}=12	v_{max}=14	v_{max}=16
WCI_{100}	100	40	4	26565.17	26356.53	26125.07	25809.67
			5	**26577.17**	26517.70	26371.07	26197.80
		50	4	26483.20	26487.40	26248.83	25967.20
			5	26546.90	26561.70	26467.50	26295.60
UCI_{500}	500	40	4	190915.27	190089.23	189437.57	186639.97
			5	190838.33	190682.17	190588.47	190443.97
		50	4	189783.00	190539.67	189216.20	186931.80
			5	189943.37	**191836.93**	191058.23	190780.77
$ISCI_{1000}$	1000	30	4	258711.50	258702.63	258066.97	257099.60
			5	258934.27	259088.20	**259114.77**	258280.57
		40	4	257657.10	258074.33	258066.87	257529.03
			5	257906.97	258804.40	258979.37	258837.37

Table 4.2: Best combinations of m and v_{max} (according to Table 4.1) in the optimization of ks_8a, ks_16b, ks_24d, WCI_{100}, UCI_{500}, and $ISCI_{1000}$.

Instance	D	m	v_{max}
Ks_8a	8	40	2
Ks_16b	16	40	2
Ks_24d	24	40	4
WCI_100	100	40	10
UCI_500	500	50	12
$ISCI$_1000	1000	30	14

quires a larger v_{max} for high-dimensional problems and a smaller v_{max} for low-dimensional problems. Note that, in practice, the original BPSO [Kennedy and Eberhart 1997] and its variants [Wang et al. 2008b, Bansal and Deep 2012, Mirjalili and Lewis 2013] used a constant v_{max} (4 or 6) instead of variable v_{max}. We conduct a logarithmic regression analysis using the dimension D and v_{max} values from Table 4.2, as shown in Fig. 4.9. Based on this analysis, we recommended the following rule for choosing v_{max} according

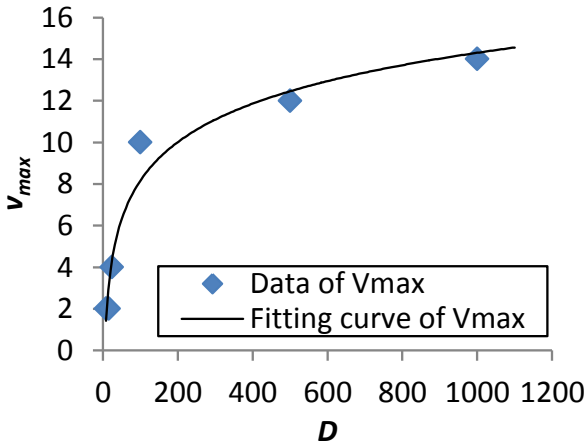


Figure 4.9: A fitting curve capturing the relationship between D and v_{max} using the logarithmic regression analysis.

to a given D value:

$$v_{max}(\mathtt{D}) = 2.6655 \ln(\mathtt{D}) - 4.10, \quad \mathtt{D} \geq 4. \tag{4.13}$$

To find out the optimal value for φ, the results on TV_T-BPSO with a fixed swarm size of 40 is presented in Table 4.3. It can be seen that TV_T-BPSO obtained the best results with φ=1 to φ=5, and thus the upper and lower limit of φ are chosen as φ_{max}=5 and φ_{min}=1, for the experiments on TV_T-BPSO in this chapter.

4.6.2 Results and Discussions

Following the previous section, we compare TV_T-BPSO with S_T-BPSO, L_T-BPSO, V_{T1}-BPSO, and V_{T2}-BPSO, respectively. For this comparison, first, we use the low-dimensional 0-1 knapsack instances, then the high-dimensional 0-1 knapsack instances. All the comparison results are averaged over 30 independent runs. For each instance, the best BPSO was compared with the other four BPSOs using a t-test with the significance level set at 0.05. If the best BPSO is significantly better than the other four, then the corresponding value is marked in bold. In addition, if a BPSO consistently achieved the optimal value or better than the best known value (where the optimal value is unknown), the corresponding entry is marked with *.

In Table 4.4, the parameter settings used for TV_T-BPSO and four other well-known BPSOs are summarized. The value of v_{max} and w of S_T-BPSO, L_T-BPSO, V_{T1}-BPSO, and V_{T2}-BPSO are set according to [Kennedy and Eberhart 1997], [Bansal and Deep 2012], [Liu et al. 2011], and [Mirjalili and Lewis 2013], respectively.

Table 4.3: Results obtained by TV_T-BPSO on Ks_8a, Ks_16b, Ks_24d, WCI_{100}, UCI_{500}, $ISCI_{1000}$ with the swarm size 40.

	Ks_8a	Ks_16b	Ks_24d
	v_{max}=2	v_{max}=2	v_{max}=4
φ=1	3921952	**9352998**	11769796.3
φ=2	**3924400**	9352998	11804350.6
φ=3	3924400	9352471.8	11806885.3
φ=4	3924400	9352998	**11810362.7**
φ=5	3924400	9352647.2	11806689.4
	WCI_{100}	UCI_{500}	$ISCI_{1000}$
	v_{max}=10	v_{max}=12	v_{max}=14
φ=1	25169.23	151145.2	242298.8
φ=2	25959.1	164316.57	256189.03
φ=3	26376.4	185893.37	257066.9
φ=4	26565.17	190089.23	258066.87
φ=5	**26577.17**	**190682.17**	**258979.37**

Table 4.4: Parameter settings of five different BPSOs for the experiments over the 0-1 knapsack benchmark problems.

Algorithm	m	c_1=c_2	v_{max}	φ_{max}	φ_{min}	w_{max}	w_{min}
S_T-BPSO	40	2	4	-	-	1	1
L_T-BPSO	40	2	4	-	-	1	1
V_{T1}-BPSO	40	2	6	-	-	0.9	0.4
V_{T2}-BPSO	40	2	6	-	-	0.9	0.4
TV_T-BPSO	40	2	Eq. (4.13)	5	1	1	1

Low-dimensional 0-1 Knapsack Instances

In this section, three different sets of 0-1 knapsack instances have been used for the comparison purpose, which are widely used test instances in the literature [Zou et al. 2011a, Bansal and Deep 2012, Liu et al. 2015b, Wang et al. 2013]. The first two sets belong to the deterministic group [Zou et al. 2011a, Wang et al. 2013] and [Bansal and Deep 2012]. Since the optimal solution of these instances is known, the comparison is made on the basis of average profit ($AvgPft$), standard deviation SD)), success rate (SR), and an average number of function evaluations ($AvgFEs$). Here, the success rate measures the percentage of successful runs where a successful run is defined as a run that produces the optimal solution before the run is terminated. For these two sets of instances, all the BPSOs in Table 4.4 will be terminated when they find the global optimum or they reach the maximum number of iterations Itr_{max}. The third set of 0-1 knapsack instances belong to the non-deterministic group. This set consists of four 100-dimensional and four 500-dimensional instances generated from the procedures in Section 3.3. More specifically, the

instances of 100 dimensions are generated using the value for R=1000, S=0.5, and n=100. Likewise, the instances of 500 dimensions are generated using the value for R=1000, S=0.5, and n=500. Since these instances have been randomly generated, the optimal solutions of these 0-1 knapsack instances are unknown. The comparison is made on the basis of average profit ($AvgPft$), standard deviation (SD), and the p-value obtained by t-test. For this set of instances, all BPSOs will be terminated when they reach the maximum number of iterations Itr_{max}.

The results on the first and second set of 0-1 knapsack instances are summarized in Table 4.5 and 4.6, respectively. The column "BK" of Table 4.6 represents the best-known profit [Bansal and Deep 2012, Liu et al. 2015b] for each of the instances of the second set.

Table 4.5 shows that TV_T-BPSO and V_{T2}-BPSO obtained similar results on the first set of 0-1 knapsack instances. However, if we compare these two BPSOs in terms of the results of SR and AFE, then TV_T-BPSO is better, since it has a higher success rate for all

Table 4.5: The obtained results on $AvgPft \pm SD$ (first line), SR (second line), and $AvgFEs$ (third line) obtained by the five BPSO variants over the test instances f_1 to f_{10} [Zou et al. 2011a, Wang et al. 2013].

Instance	D	Opt	TV_T-BPSO*	BPSO	L-BPSO	V_{T1}-BPSO	V_{T2}-BPSO
f_3	4	35	35.00±0.00 100 **41**	35.00±0.00 100 92	35.00±0.00 100 45	34.70±1.32 93.33 2712	35.00±0.00 100 43
f_4	4	23	23.00±0.00 100 **41**	23.00±0.00 100 461	22.97±0.18 96.67 1372	22.97±0.18 96.67 1372	23.00±0.00 100 43
f_9	5	130	130.00±0.00 100 **54.67**	130.00±0.00 100 1058.67	127.20±5.16 76.67 9373.33	128.40±4.15 86.67 5377.33	130.00±0.00 100 56
f_7	7	170	107.00 ±0.00 100 **140**	103.90 ±4.28 30 28019	105.73±1.41 46.67 21364	103.53±4.54 43.33 22685	107.00±0.00 100 167
f_1	10	295	295.00±0.00 100 **413**	294.57±1.38 86.67 8556	290.77±10.66 56.67 17429	281.83±21.11 26.67 29369	295.00±0.00 100 851
f_6	10	52	52.00 ±0.00 100 **180**	51.80±0.55 86.67 7913	51.63±0.85 83.33 6973	51.33±1.15 66.67 13469	52.00±0.00 100 280
f_5	15	481.07	481.07±0.00 100 **1976**	478.19±10.94 93.33 4223	474.80±17.01 73.33 10920	427.19±39.87 16.67 33364	481.07±0.00 100 14715
f_2	20	1024	1024.00±0.00 100 **6820**	1024.00±0.00 100 3533	1002.53±22.32 26.67 29449	931.27±53.80 0 40000	1024.00±0.00 100 31289
f_{10}	20	1025	1025.00±0.00 100 4707	1025.00±0.00 100 **1980**	989.70±36.43 13.33 34703	933.07±50.78 3.33 38681	1025.00±0.00 100 29440
f_8	23	9767	9767.00±0.00 **100** 11060	9766.93±0.37 96.67 6001	9766.30±2.59 90 6424	9757.33±8.93 20 32371	9766.97±0.18 96.67 36060

Table 4.6: The results on $AvgPft$ (first line), SD (second line), SR (third line), $AvgFEs$ (fourth line) of five different BPSOs over the test instances Ks_16a to Ks_24e [Bansal and Deep 2012].

Instance	D	Opt.	BK	TV_T-BPSO*	S_T-BPSO	L_T-BPSO	V_{T1}-BPSO	V_{T2}-BPSO
Ks_16a	16	7850983	7848800	7850983.00 ±0.00 100 **9252**	7771152.37 ±76092.77 16.67 35263	7797246.87 ±67539.40 26.67 29888	7693936.93 ±122910.45 16.67 33775	7850983.00 ±0.00 100 21676
Ks_16b	16	9352998	9352800	9352998.00 ±0.00 100 **7835**	9245414.87 ±84778.02 26.67 30153	9250490.53 ±120292.36 26.67 29907	9127967.97 ±159529.14 3.33 38725	9352998.00 ±0.00 100 19859
Ks_16c	16	9151147	9149200	9150326.30 ±4495.16 96.67 **9223**	9036801.60 ±105374.18 13.33 35448	9052814.63 ±83568.41 16.67 33597	8924175.00 ±147544.37 6.67 37441	9150326.30 ±4495.16 96.67 19669
Ks_16d	16	9348889	9345000	9347669.20 ±3721.96 **90** **13440**	9271454.97 ±69041.39 13.33 35053	9294715.13 ±52798.58 20 32740	9188651.17 ±95956.06 0 40000	9347262.60 ±4217.41 86.67 25804
Ks_16e	16	7769117	7767300	**7768496.13** **±3400.63** **96.67** **13660**	7693213.00 ±75474.04 13.33 35517	7713892.53 ±78597.11 30 28467	7587932.27 ±125788.08 3.33 38681	7767875.27 ±4725.57 93.33 21015
Ks_20a	20	10727049	10720314	**10724840.50** **±7523.13** **90** **18976**	10669618.47 ±60759.13 26.67 32543	10692481.67 ±58716.61 33.33 27853	10563639.87 ±120918.17 3.33 38672	10714666.40 ±13060.22 43.33 38615
Ks_20b	20	9818261	9805480	**9815420.10** **±9323.38** **90** **18389**	9735538.37 ±79571.34 20 34121	9766711.50 ±57579.60 30 29204	9632312.53 ±112517.84 0 40000	9797837.57 ±20116.05 40 38263
Ks_20c	20	10714023	10710947	**10712635.83** **±4077.25** **86.67** **19208**	10615440.83 ±115323.68 23.33 32559	10688081.47 ±45556.95 36.67 26319	10486056.43 ±133805.32 0 40000	10709990.30 ±6230.09 60 36317
Ks_20d	20	8929156	8923712.21	**8929156.00** **±0.00** **100** **15155**	8866124.10 ±74952.33 33.33 31507	8883763.17 ±70473.79 43.33 24383	8735162.43 ±104543.78 0 40000	8916392.47 ±15587.49 53.33 37295
Ks_20e	20	9357969	9355930.35	**9356953.67** **±3038.76** **66.67** **24236**	9306943.83 ±54249.45 23.33 34183	9318815.13 ±68243.98 20 32973	9181466.33 ±126241.20 0 40000	9353699.90 ±9589.14 56.67 36537
Ks_24a	24	13549094	13532060	**13533425.33** **±20555.05** **56.67** **30016**	13507548.40 ±58236.73 36.67 31949	13512451.77 ±31391.28 26.67 31051	13323722.73 ±137591.62 3.33 38796	13499799.43 ±22346.71 0 40000
Ks_24b	24	12233713	12223443	**12222004.63** **±15052.33** **53.33** **29381**	12152367.53 ±71669.10 16.67 36640	12191466.07 ±51448.35 30 30167	12031294.90 ±107235.86 0 40000	12182354.87 ±25839.01 3.33 39867
Ks_24c	24	12448780	12443349	**12444101.20** **±10802.00** **70** 29713	12395398.53 ±78169.05 36.67 31723	12420324.73 ±47375.30 40 26820	12255888.23 ±114864.01 3.33 39008	12414359.70 ±27122.87 16.67 38471
Ks_24d	24	11815315	11803712	**11809811.83** **±9023.92** **50** 32332	11778345.20 ±58548.76 46.67 29028	11776442.47 ±59364.89 26.67 30903	11632526.57 ±128661.51 0 40000	11776692.07 ±21915.93 10 39400
Ks_24e	24	13940099	13932526	**13937049.83** **±6653.97** **76.67** 28019	13914845.70 ±45334.60 43.33 30343	13904779.23 ±67627.36 43.33 25769	13777674.00 ±150279.17 13.33 35296	13897801.70 ±26691.06 13.33 39533

the 0-1 knapsack instances with the least amount of AFE. This proves that TV_T-BPSO is reliable and faster than the four other BPSOs over the first set of 0-1 knapsack instances. Compared to TV_T-BPSO, V_{T1}-BPSO has the lowest success rate and the highest amount of AFE for all the instances, except the instance ks_24d. Therefore, it can be said that V_{T1}-BPSO is not a reliable neither a faster algorithm for this set of instances.

Table 4.6 shows that TV_T-BPSO performs well on the optimization of the second set of 0-1 knapsack instances. In particular, TV_T-BPSO obtained better results than the best-known results and the results obtained by the other four BPSOs with the least amount of $AvgFEs$. In addition, TV_T-BPSO obtained these results with the highest success rate. These results show that TV_T-BPSO is more reliable and faster for this set of instances. Table 4.6 also shows that V_{T2}-BPSO obtained similar results to those of TV_T-BPSO on the instances of 16-D. However, the performance of V_{T2}-BPSO decreases significantly with the increase of dimensions of the instances Ks_20a to Ks_24e. It can be seen that V_{T1}-BPSO shows the worse performance in terms of $AvgPft$, SD, SR, and $AvgFEs$.

Table 4.7 summarized the results on 8 non-deterministic 0-1 knapsack instances of 100 and 500 dimensions. In this table, the first and second lines have been used to present $AvgPft \pm SD$ and the p-values obtained by t-test, respectively. A highlighted p-value (< 0.05) means that the results obtained by TV_T-BPSO are much better than the results obtained by the existing BPSOs or vice verse. The last row of this table represents the number of instances that TV_T-BPSO wins/ties/loses over the existing BPSOs.

Table 4.7 shows that TV_T-BPSO significantly performs well on the optimization of

Table 4.7: The results on $AvgPft \pm SD$ (first line) and p-value (second line) of five different BPSOs over 100-D and 500-D non-deterministic 0-1 knapsack instances. Under the t-test, if TV_T-BPSO is statistically better than an existing BPSO, then the p-value corresponding to that BPSO is highlighted.

Instance	D	TV_T-BPSO*	S_T-BPSO	L_T-BPSO	V_{T1}-BPSO	V_{T2}-BPSO
UCI_{100}	100	41082.83±40.06 -	40712.53±220.53 **3.38E-10**	40676.07±692.24 **0.003193397**	38131.00±1802.62 **7.31E-10**	34830.10±534.03 **4.75E-33**
WCI_{100}	100	26656.13±29.68 -	25537.77±172.07 **2.37E-26**	26441.40±90.42 **2.34E-14**	26410.83±127.55 **1.15E-11**	25243.77±116.74 **5.13E-36**
SCI_{100}	100	30499.50±40.15 -	29684.67±124.12 **1.71E-28**	30257.10±80.75 **2.70E-18**	30355.13±148.75 **1.23E-05**	29507.93±90.99 **4.20E-39**
$ISCI_{100}$	100	23215.43±37.03 -	22345.37±115.84 **2.02E-30**	22921.5±101.12 **3.57E-17**	23038.53±148.30 **3.76E-07**	22182.30±84.09 **3.95E-41**
UCI_{500}	500	194552.37±599.16 -	149200.87±2716.56 **9.09E-40**	175225.43±3178.31 **1.22E-25**	173451.57±11976.25 **1.44E-10**	141932.70±1510.21 **5.89E-57**
WCI_{500}	500	135212.10±230.51 -	128062.43±375.87 **5.26E-55**	129691.47±250.54 **2.48E-63**	134244.50±511.36 **8.92E-12**	127765.13±257.35 **3.98E-70**
SCI_{500}	500	150899.93±204.06 -	145025.33±279.12 **1.65E-60**	146495.30±279.03 **6.24E-54**	150448.47±458.71 **1.50E-05**	144913.63±178.81 **2.06E-70**
$ISCI_{500}$	500	130218.50±196.51 -	125561.10±212.23 **3.31E-63**	126668.97±204.35 **3.75E-57**	129927.20±325.56 **0.000117656**	125454.73±157.00 **4.67E-65**
w/t/l		-	8/0/0	8/0/0	8/0/0	8/0/0

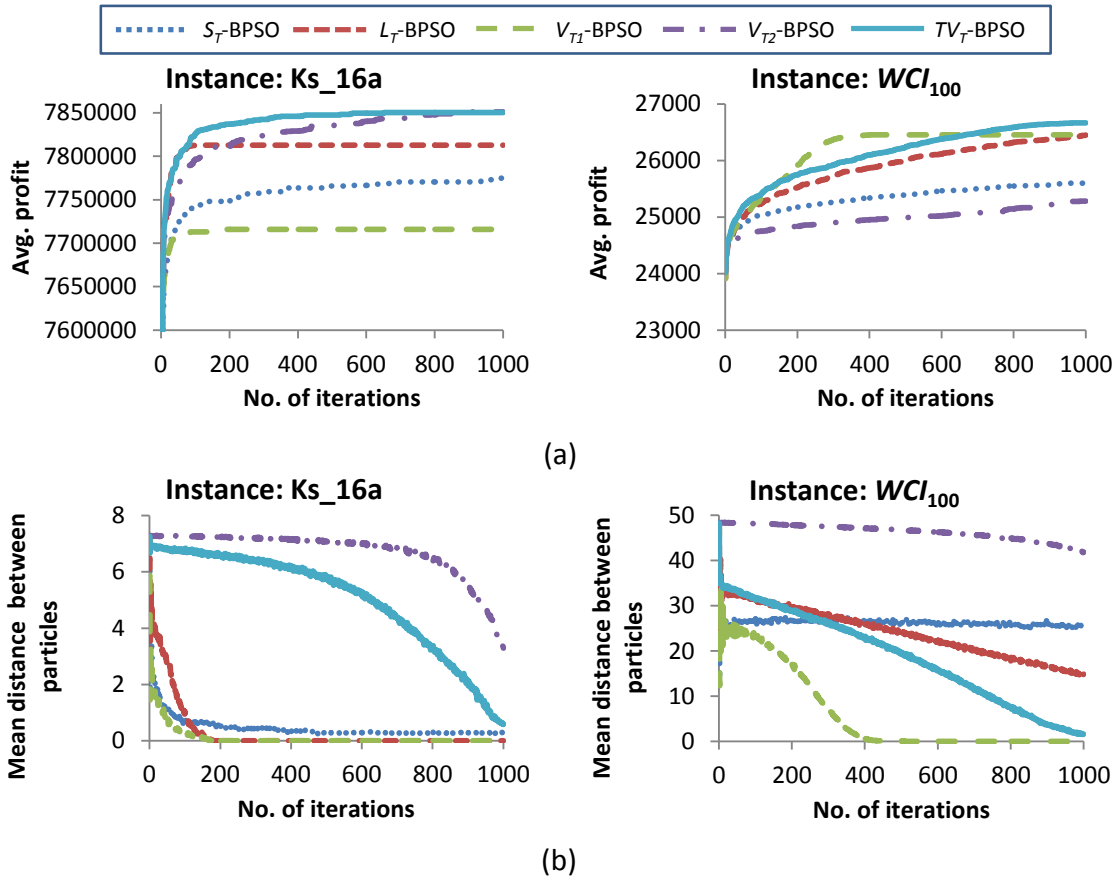


Figure 4.10: Comparing five different BPSOs on the benchmark instances ks_16a and WCI_{100} on their a) convergences, and b) diversity profiles.

100-D, 500-D non-deterministic instances. In contrast, V_{T2}-BPSO shows the worst performance on the optimization of this set of test instances. Specifically, the performance of V_{T2}-BPSO has been decreased with the number of dimensions of the problems. It is mentioned in [Mirjalili and Lewis 2013], V_{T2}-BPSO significantly performs well compared to the well-known BPSOs [Lee et al. 2008, Shen et al. 2004, Wang et al. 2008a] on the set of low-dimensional continuous optimization problems [Suganthan et al. 2005]. However, our experimental results show that V_{T2}-BPSO is not suitable for the higher dimensional problems. In this set of instances, V_{T1}-BPSO obtained the better optimization results compared to S_T-BPSO and L_T-BPSO.

Figure 4.10 shows the convergence and diversity profiles of the five BPSO variants on the instances ks_16a and WCI_{100}. From Fig. 4.10(a), it can be seen that TV_T-BPSO was able to outperform the other BPSO variants consistently on both the deterministic and non-deterministic 0-1 knapsack benchmark instances. V_{T1}-BPSO performed poorly ks_16a. However, though better on WCI_{100} of 100-D. Comparing to V_{T1}-BPSO, V_{T2}-BPSO performed better on the instance ks_16a but poorly on the instance WCI_{100}.

Fig. 4.10(b) shows the diversity profiles on instances ks_16a and WCI_{100}. This figure shows that TV_T-BPSO can maintain a better diversity for both instances resulting in a good balance between exploration and exploitation during the run. It can be noted that the swarm diversity decreases to zero towards the end of the run, which shows that TV_T-BPSO can provide a stronger exploitation in the final stage. This helps explain the better results by TV_T-BPSO on the above sets of 0-1 knapsack instances. In contrast, Fig. 4.10(b) shows that S_T-BPSO, L_T-BPSO, V_{T1}-BPSO, and V_{T2}-BPSO are unable to maintain a good balance between exploration and exploitation, resulting in their poorer results.

High-dimensional 0-1 Knapsack Instances

Table 4.5 to Table 4.7 show that TV_T-BPSO performs well on instances up to 500 dimensions. In order to investigate the scalability of TV_T-BPSO to even higher dimensions, additional experiments have been carried out on twelve 0-1 knapsack instances of 1000, 2000, and 5000 dimensions. For these experiments, we use the same parameter settings as shown in Table 4.4.

Table 4.8 presents the results by the five different BPSOs on 1000-D, 2000-D, and 5000-D instances, where the last row represents the number of instances that TV_T-BPSO

Table 4.8: The results on $AvgPft \pm SD$ (first line) and p-value (second line) of five different BPSOs over 1000-D to 5000-D non-deterministic 0-1 knapsack instances.

Instance	D	TV_T-BPSO*	S_T-BPSO	L_T-BPSO	V_{T1}-BPSO	V_{T2}-BPSO
UCI_{1000}	1000	380640.03±1397.89 -	285096.00±2907.05 **4.53E-60**	312836.60±4455.70 **8.09E-41**	340616.60±19044.61 **2.29E-12**	276968.53±2047.78 **2.09E-75**
WCI_{1000}	1000	269081.53±578.08 -	256849.47±628.30 **2.94E-60**	259007.63±391.60 **5.33E-55**	268128.43±1091.88 **0.000118**	256723.57±296.24 **1.42E-53**
SCI_{1000}	1000	312629.20±520.02 -	303167.17±429.02 **2.08E-55**	305246.57±420.93 **2.79E-50**	312332.83±775.88 0.099603	303130.40±283.43 **9.66E-50**
$ISCI_{1000}$	1000	262444.47±525.65 -	254392.30±333.35 **4.28E-51**	256063.80±319.78 **1.36E-45**	262017.30±630.85 **0.006114**	254470.63±374.25 **1.12E-52**
UCI_{2000}	2000	756871.80±3380.62 -	559550.23±5350.69 **1.24E-69**	598293.10±6658.42 **2.14E-55**	660994.97±32962.45 **5.27E-16**	550112.60±3337.33 **5.27E-16**
WCI_{2000}	2000	530579.83±751.70 -	512181.07±695.33 **6.32E-66**	515252.33±795.73 **7.39E-60**	529786.27±1093.49 **0.001892**	512122.00±520.41 **5.30E-63**
SCI_{2000}	2000	624930.57±898.16 -	611012.67±491.29 **1.04E-48**	613746.70±597.27 **2.14E-47**	625499.53±1073.54 **0.030007**	611051.63±429.11 **2.45E-46**
$ISCI_{2000}$	2000	516189.63±785.10 -	504070.47±503.50 **1.90E-51**	506166.50±462.34 **4.01E-46**	516505.00±735.64 0.113843	503910.47±371.00 **2.25E-46**
UCI_{5000}	5000	1758217.77±7747.40 -	1327171.07±7616.02 **4.09E-86**	1385014.63±9786.66 **9.39E-76**	1527048.27±48696.11 **3.35E-22**	1311025.30±3667.60 **8.20E-70**
WCI_{5000}	5000	1302697.10±1494.24 -	1268726.00±964.20 **7.84E-60**	1273625.77±1047.98 **4.89E-58**	1299083.80±2204.27 **1.13E-09**	1268303.47±666.19 **3.70E-52**
SCI_{5000}	5000	1283927.03±1314.99 -	1250033.87±1000.27 **7.72E-66**	1255255.63±756.80 **1.90E-56**	1281307.33±1937.18 **1.29E-07**	1249858.17±540.63 **1.06E-52**
$ISCI_{5000}$	5000	1262279.00±1304.29 -	1239822.40±768.42 **3.58E-52**	1243482.67±714.20 **2.52E-47**	1262350.17±1562.29 0.848801	1240059.40±678.69 **1.30E-49**
w/t/l		-	12/0/0	12/0/0	12/3/0	12/0/0

wins/ties/loses. It can be observed that TV_T-BPSO significantly outperformed other four BPSO variants on the non-deterministic 0-1 knapsack instances with more than 1000 dimensions. In contrast, the performance of V_{T2}-BPSO deteriorates a great deal on this set of instances. It is also worth noting that for this set of instances, V_{T1}-BPSO performed better than S_T-BPSO, L_T-BPSO, and V_{T2}-BPSO.

4.7 Chapter Summary

In this chapter, we have proposed a modified version of BPSO termed TV_T-BPSO, which adopts a time-varying transfer function to address the shortcoming of existing transfer functions by providing a better balance between exploration and exploitation for the BPSO during its optimization run. We have presented some empirical analyses of the search behaviours of these BPSO variants using different transfer functions, in an effort to understand how this time-varying transfer function makes a good balance between exploration and exploitation during the search. The behaviours of TV_T-BPSO are compared with that of the original BPSO (S_T-BPSO) and three other well-known modified BPSOs over several low-dimensional (D$\leq$500) 0-1 knapsack benchmark instances. Our experimental results have demonstrated that TV_T-BPSO outperforms the existing BPSOs over these low-dimensional benchmark instances. Furthermore, we have examined the scalability of TV_T-BPSO on high-dimensional (1000 $\geq$D$\leq$5000) 0-1 knapsack benchmark instances. The experimental results have shown that TV_T-BPSO can scale well to high-dimensional combinatorial optimization problems.

In the following chapter, we will deal with a real-world structural optimization problem namely the truss optimization problem. For the truss design problem, we will develop a bilevel niching method where the proposed TV_T-BPSO will be used along with the B-SPSO for the truss topology optimization.

CHAPTER 5

Multimodal and Bilevel Techniques for Truss Design Problems

5.1 Introduction

Truss design is a well-known structural optimization problem which has important practical applications in various engineering fields [Naceur et al. 2004, Huang and Xie 2010, Zhou et al. 2010, Sutradhar et al. 2010]. Truss design problems are typically multimodal by nature, meaning that it offers multiple optimal solutions with respect to the topology and/or sizes of the members, but they are evaluated to have similar or equally good objective function values. From a practical standpoint, it is desirable to find as many alternative designs as possible, rather than finding a single design, as often practiced. To facilitate this, this chapter proposes a bilevel formulation for the truss design problem so that multiple truss topologies and their size solutions can be obtained in the simultaneous manner. This formulation is the precursor to the development of a bilevel niching method which is developed based on the B-SPSO (Chapter 3), TV_T-BPSO (Chapter 4), and a standard PSO (see Chapter 2 Section 2.3.2). The proposed bilevel niching method has the ability to locate multiple topologies and their corresponding size solutions of well-known truss design problems.

The rest of the chapter is organized as follows. Section 5.2 provides the motivation of this chapter. Section 5.3 presents the related work. Section 5.4 describes the bilevel formulation of the truss optimization problem. Section 5.5 describes the proposed bilevel niching method. In Section 5.6, some challenging low- and high-dimensional truss design problems are selected from literature, on which the accuracy, robustness, and efficiency of the proposed bilevel niching method are evaluated and compared with the best available methods in the literature. The concluding remarks of this chapter are provided in Section 5.7.

5.2 Motivation

The optimal design of truss structures using metaheuristic methods has been an active area of research since it was initiated by the work of Goldberg and Samtani [Goldberg and Samtani 1986]. Since then, several metaheuristic methods have been proposed for truss optimization. While most of these metaheuristics deal with the size optimization of the truss structures [Adeli and Kamal 1991, Flager et al. 2014, Li et al. 2007, Kaveh et al. 2014, Ho-Huu et al. 2016], metaheuristic methods have also been proposed for optimizing simultaneously the topology, and size of the truss structures [Deb and Gulati 2001, Tang et al. 2005, Fenton et al. 2016, Ahrari and Deb 2016]. It has been shown that the simultaneous optimization of the topology, and size are more accurate than the just the size optimization [Deb and Gulati 2001]. Nevertheless, these metaheuristic methods have been mostly designed for finding a single optimal solution instead of multiple optimal solutions for a specific truss problem.

Niching methods, which are purposely designed for finding multiple solutions in a single optimization run [Goldberg and Richardson 1987], can be effectively used for finding multiple truss design solutions. Recently, a few research works [Luh and Lin 2008; 2011, Li 2015] adopted niching methods in the popular single- or two-stage truss optimization approaches for finding multiple topologies and their corresponding size solutions. Basically, the two-stage approach uses a niching technique in the first stage to identify multiple truss topologies. In this case, a fixed cross-sectional area for all the members of a given ground structure is considered for the evaluation purpose. In the second stage, a standard optimizer is used to find the optimal/near optimal size solution for all identified topologies. It is noticeable that the two-stage approach optimizes the topology and size in two different stages with the assumption that these two sub-problems are linearly separable. However, in reality, these two problems are non-separable [Deb and Gulati 2001]. For example, a topology found in the first stage might be feasible for a given fixed cross-sectional area of the members, but this may not be feasible in the second stage for different cross-sectional areas of the members due to the stress and displacement constraints. Therefore, the optimal designs derived from a given ground structure may not be attainable with the two-stage approach. In case of the single-stage approach, the topology and size optimization are performed together in the following manner: firstly the single-stage approach uses a niching technique to perform the size optimization of the given ground structure. Then, non-active members are identified (whose cross-sectional area found to be zero or less than the critical area ϵ) and eliminated to realize the topology from the given ground structure. It is noticeable that in this approach, the optimal/near optimal topologies highly depend on the found size solutions.

From the above observations, we can see that it would be desirable to overcome these shortcomings of the single- and two-stage truss optimization approaches.

5.3 Related Work

In the past decades, several techniques based on classical optimization methods have been developed for simultaneous optimization of *topology, size*, and *shape* of a truss structure [Ringertz 1985, Kirsch 1989]. Metaheuristic methods have also been adopted for this purpose because of their appealing properties, i.e., they do not make certain assumptions and possess better global search capabilities than the classical methods. For example, a real-coded Genetic Algorithm (GA) scheme was introduced in [Deb and Gulati 2001] to efficiently handle the truss optimization problems, where the GA operators are directly applied to real-value coded variables instead of binary strings. Mixed encoding schemes including binary, real numbers, and integers are also employed by the GA to encode the variables of a truss problem [Tang et al. 2005]. This encoding scheme, so called *surrogated reproduction* is adopted for producing offspring from the parent solutions.

Many other metaheuristic methods including evolutionary algorithms (EAs), differential evolution (DE), evolutionary strategy (ES), firefly algorithm (FA), and particle swarm optimization (PSO) have been used for the simultaneous optimization of the topology, and size of the truss problem [Ahrari and Deb 2016, Noilublao and Bureerat 2011, Miguel et al. 2013]. The optimization results show that DE, ES, FA, and PSO can produce better quality solutions compared to the GA based methods [Ahrari and Deb 2016, Ahrari et al. 2015, Wu and Tseng 2010, Wu et al. 2017]. Nevertheless, it can be observed that all these methods were designed for obtaining a single solution instead of multiple solutions for a truss optimization problem in a single optimization run.

Niching methods are well-suited optimization methods for finding multiple optimal solutions in a single optimization run [Goldberg and Richardson 1987]. Classic niching methods include fitness sharing [Goldberg and Richardson 1987], clearing [Petrowski 1996], speciation [Li et al. 2002, Li 2004]. Niching methods were first adopted in [Hajela et al. 1993] for multimodal structural optimization, where a GA based fitness sharing method is used to find multiple topologies for a truss structure. Later on, in [Luh and Lin 2008], an ant algorithm is combined with fitness sharing for the same purpose. In [Luh and Lin 2011], the fitness sharing concept is incorporated into a modified binary PSO for multimodal truss optimization. In these methods [Hajela et al. 1993, Luh and Lin 2008; 2011], the truss topology and size variables are optimized in two different stages where the topology and size variables are assumed to be linearly separable. In such a multi-stage approach, different topologies are obtained in the first stage by assuming an equal cross-section area for each member of a ground structure. In the later stage, the area of each member of these obtained topologies is optimized to realize the optimal design of a truss problem. It is apparent that the true optimal/near-optimal designs of a truss structure may not be achievable by such a method, since these two types of variables are not necessarily linearly separable [Deb and Gulati 2001]. Recently, Li [Li 2015] proposed an improved species-convergence genetic algorithm (SCGA) for the multimodal truss optimization problems.

In this method, the niching technique is applied to the single-stage truss optimization method for multiple topologies and their corresponding size solutions. In such a case, the truss topologies are determined based on the size solutions of a given ground structure [Deb and Gulati 2001]. It is obvious that such a method may not obtain the optimal/near optimal truss design solutions, because the derived topologies are highly depends on the found size solutions of the ground structure. This motivates us to introduce a new bilevel formulation for the truss problem which is elaborated in the following section.

5.4 Problem Formulation

The goal of this research is to find multiple design solutions for the truss design problems in terms of the topology and size. To do this, the truss topology and size optimization[1] need to be performed simultaneously [Wang et al. 2004], not separately, as often practiced [Hajela et al. 1993, Luh and Lin 2008; 2011]. Following this, we formulate the truss design problem as a bilevel optimization problem[2]. The bilevel formulation contains two levels of optimization tasks where the size optimization task is nested within the topology optimization. The nested structure of the overall problem requires that a solution to the topology problem may be feasible only if it is an optimal solution to the size problem.

We begin the bilevel formulation for a truss problem by considering its ground structure, which is a complete truss with all possible member (M) connections among all nodes (N) in the structure, as shown in Fig. 5.1(a). For the bilevel formulation, the topology variables of such a ground structure are treated as the variables of the upper-level problem, and at the same time, size variables are treated as the variables of the lower-level problem. Considering this, the objective functions of the upper-level problem and lower level problem are provided below.

Upper level problem: For a given set of members M and nodes N of a ground structure (see Fig. 5.1(a)), the upper level optimization task is to select a subset of members to find a stable topology subject to the given constraints. In this case, we consider $x_u \in \mathbf{x}_u$ as a upper level problem which represents a stable topology of the ground structure consisting of m members and n nodes (see Fig. 5.1(b)). Here, the variables that are related to x_u are the discrete variables. Now, the lower level optimization task is to find the optimal cross-sectional areas of the members of topology x_u (see Fig. 5.1(c)). Let $x_l \in \mathbf{x}_l$ be the lower level problem consisting of the set of member cross-sectional areas $\mathbf{A}$ and the set of nodal coordinates ξ of topology x_u. Here, the variables that are related to x_l are the continuous variables. Considering the upper level variable x_u and lower level variable x_l,

[1] The more generic background information on truss topology and size optimization can be found in Chapter 2.

[2] The more generic background information on bilevel optimization is provided in Chapter 2.

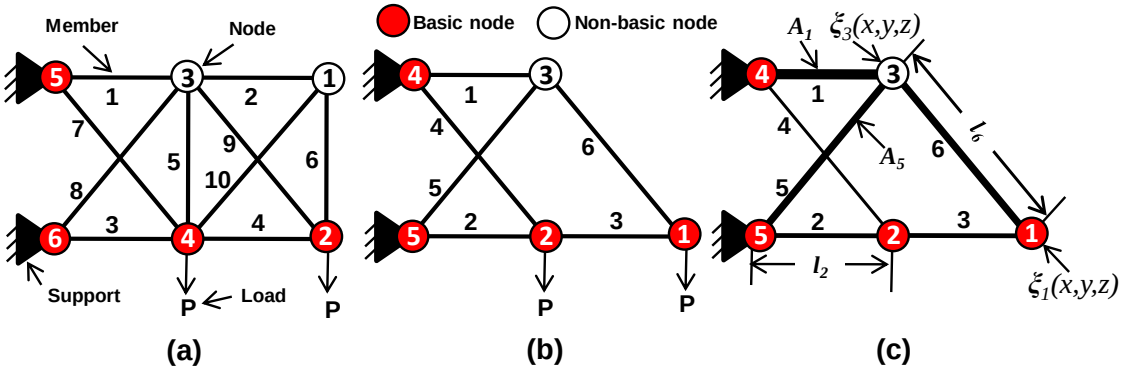


Figure 5.1: Illustration of (a) the 10-member ground structure, (b) one of its possible topology solutions, and (c) one of the possible size solutions of that topology where A represents the member cross-sectional area, l represents the member length, and $\xi(x, y, z)$ represents the node coordinate.

the objective function of the upper level problem is formulated as:

$$\begin{aligned} &\text{find} \quad x_u \\ &\min_{x_u \in \mathbf{x}_u, x_l \in \mathbf{x}_l} W(x_u, x_l) \\ &\quad \text{s. t. } G_1(x_u, x_l) : \textit{Truss contains all the basic nodes}, \\ &\qquad G_2(x_u, x_l) : \textit{Truss is internally and externally stable}, \\ &\qquad G_3(x_u, x_l) : x_l \in \operatorname*{argmin}_{x_l \in \mathbf{x}_l} \{w(x_u, x_l) : g_j(x_u, x_l) \\ &\qquad\qquad \leq 0,\ j = 1 \ldots J\}. \end{aligned} \tag{5.1}$$

where W represents the weight of the topology x_u, constraint G_1 ensures that the truss consists of all the basic nodes, and constraint G_2 ensures the internal and external stability of the truss. The internal and external stability of a truss can be checked by the following Grubler's inequality equations [Ghosh and Mallik 2011]:

$$m \geq 2n - 3, \tag{5.2}$$

and

$$m + r \geq 2n, \tag{5.3}$$

where m denotes the number of members exist in x_u, n denotes the number of nodes exist in x_u, and r denotes the number of reaction components [Krenk and Høgsberg 2013]. Finally, constraint G_3 associates with the lower level problem which is described below.

Lower level problem: In a bilevel truss problem, the lower level task is associated with the size optimization of a topology provided by the upper level, subject to the given

constraints. Specifically, the goal of the lower level optimization is to minimize the weight of a topology solution x_u (received from the upper level) by optimizing its member cross-sectional areas. To achieve this goal, the lower level problem is expressed as a nonlinear programming problem (NLP) in the following way:

$$\begin{aligned} \min_{x_l \in \mathbf{x}_L} \; & w(x_u, x_l) = \sum_{i=1}^{m} \rho_i \ell_i A_i \\ \text{s. t.} \; & g_1(x_u, x_l) : \textit{Truss is kinematically stable} \\ & g_2(x_u, x_l) : S_i \geq \sigma_i(\mathbf{A}, \xi), \; i = 1, 2, .., m, \\ & g_3(x_u, x_l) : \delta_j^{max} \geq \delta_j(\mathbf{A}, \xi), \; j = 1, 2, .., n, \\ & g_4(x_u, x_l) : A_{min} \leq A_i \leq A_{max}, \; i = 1, 2, .., m, \\ & g_5(x_u, x_l) : \xi_{min} \leq \xi_j \leq \xi_{max}, \; j = 1, 2, .., n, \end{aligned} \tag{5.4}$$

where m and n represent the number of members and nodes of the topology x_u, respectively. The parameters ρ_i and ℓ_i are the material density and length of the i-th member of the topology x_u, respectively. Here, the length ℓ_i depends on the start coordinate $\xi_i^s(x, y, z) \in \xi$ and end coordinate $\xi_i^e(x, y, z) \in \xi$ of the i-th member of x_u topology. For example, the length of the 6^{th} member l_6 (see Fig. 5.1(c)) depends on its two coordinates ξ_1 and ξ_3. The parameter $A_i \in \mathbf{A}$ represents the cross-sectional area of the i-th member of topology x_u. In this equation, constraint g_1 ensures that the kinematical stability of the truss is checked by determining the positive definiteness of the stiffness matrix. Constraint g_2 ensures that stress σ_i of a member is less than or equal to the allowable stress S_i. Constraint g_3 ensures that the displacement δ_j of j-th node is less than or equal to the allowable displacement δ_j^{max}. Constraint g_4 ensures that the value of A_i is within the limits $[A_{min}, A_{max}]$. Finally, constraint g_5 is used to ensure that the coordinates of the j-th node ξ_j is within the limits $[\xi_{min}, \xi_{max}]$.

5.5 Bilevel Niching Method for Truss Optimization

The bilevel truss problem formulated in the previous section can be used to obtain multiple design solutions in terms of topology as well as the sizes of cross-sectional areas. Although it is possible to apply niching at both the upper and lower levels, applying niching to the upper level alone seems to be sufficient to provide the necessary diversity in topology and size solutions. We choose to use a standard meta-heuristic for the lower level optimization so as to keep the proposed bilevel niching method as a whole more computationally tractable. Fig. 5.2 provides an example to illustrate this. Here, niching is applied only to the upper level. In the upper level, initially a number of individuals (representing different topologies, e.g., topologies A, B, and C) are randomly generated. If these topologies are found to be valid, then they will be sent to the lower level for size optimization. In Fig. 5.2, the topologies after the size optimization are represented by

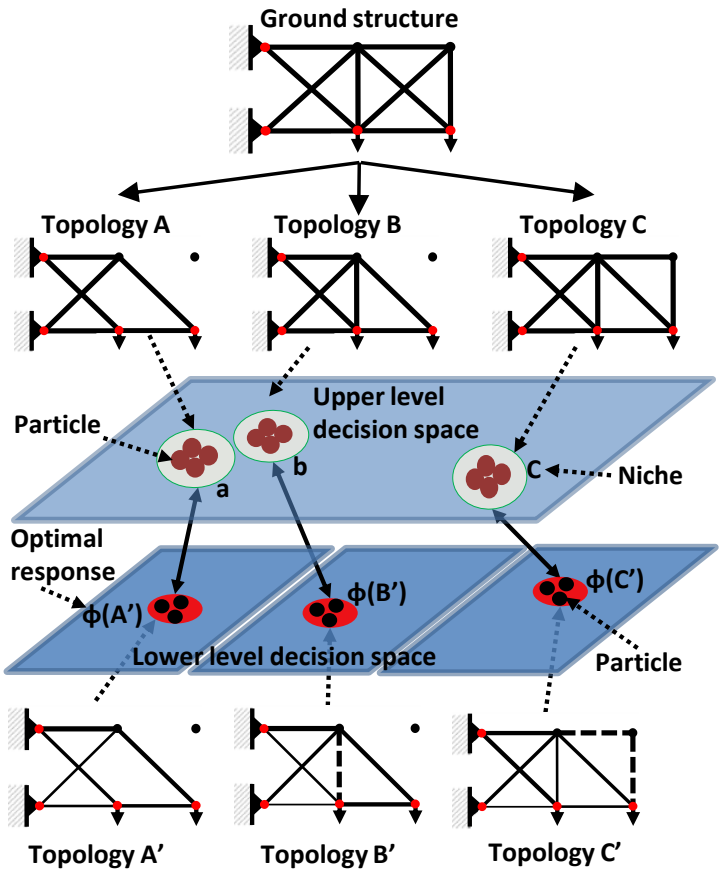


Figure 5.2: Illustration of an example where niching is applied in the upper level and a standard optimizer is used at the lower level to optimize a bilevel truss problem.

solutions A', B', and C', where the dashed lines represent the non-active members[3] of these topology solutions. After the size optimization, all these candidate solutions will be sent back to the upper level optimizer. At the upper level, niching is applied to these topology solutions in order to form different species. The best topology of each specie will be chosen to survive to the next iteration. Now consider that A', B', and C' are the best solutions found in the final stage of the run. According to [Deb and Gulati 2001], if we remove the non-active members from solutions A', B', and C', then some solutions (e.g., A' and B') may be found to have the same topology, but there is a high probability that their size solutions are still different from each other, since they are obtained through optimization in different species.

The proposed method achieves niching at the upper level by adopting a binary niching method, more specifically an updated version of B-SPSO. At the lower level, a basic PSO is used [Shi and Eberhart 1998]. This bilevel niching method encourages good and also

[3]If the cross-sectional area (A_i) of a truss member found to be less than the critical cross-sectional area (ϵ) i.e., $A_i < \epsilon$, then this type of truss member is called non-active member.

different topological solutions to be identified without imposing too early the sizing and other constraints. Similarly good feasible size solutions (of different topologies) found from the lower level would also have a chance to participate in the niching process at the upper level. Each iteration of the upper and lower level optimization continues to improve the overall quality of the solutions. By allowing a more diverse pool of solutions optimized at both upper and lower level, we avoid making the assumption that topology and size are two separable aspects of the truss problem. In the following section, we will describe the modified binary SPSO before introducing the working steps of the proposed bilevel niching method.

5.5.1 The Modified Binary SPSO (MB-SPSO) Niching Method

In Chapter 3, we developed a binary niching method (termed as B-SPSO) by adopting the speciation concept [Li et al. 2002, Li 2004] in binary PSO (BPSO). The experimental results showed that this B-SPSO has the difficulties of providing the satisfactory results specially for the high-dimensional 0-1 knapsack test instances. Our investigation in Chapter 4 shows that the standard BPSO has the difficulties in maintaining a good balance between the exploration and exploitation which restrict B-SPSO from providing satisfactory results for the high-dimensional 0-1 knapsack test instances of Chapter 3. To overcome the limitation of BPSO, Chapter 4 introduced a time-varying parameter φ into the sigmoid function in a modified *gbest* BPSO (termed as TV_T-BPSO), creating a new time-varying transfer function (TV_T), as illustrated in Fig. 5.3. The curves in Fig. 5.3

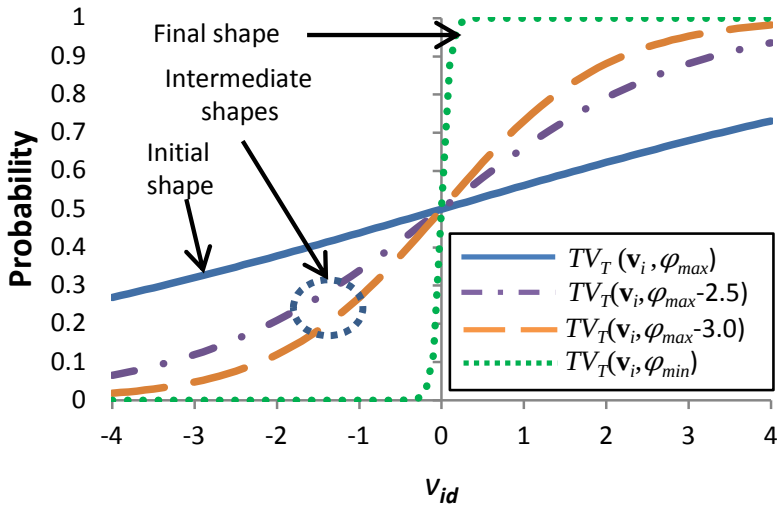


Figure 5.3: Illustration of the time-varying sigmoid transfer function (TV_T).

are produced by this time-varying sigmoid transfer function TV_T as defined below:

$$TV_T(\mathbf{v}_i^{k+1}, \varphi) = \frac{1}{1 + e^{-\frac{\mathbf{v}_i^{k+1}}{\varphi}}}, \tag{5.5}$$

where $\mathbf{v}_i^{k+1}$ is the velocity of the i-th particle at $(k+1)$-th iteration; φ is a time starting with a large value (φ =4) and gradually decreased to (φ=0.5) as the run progresses.

For this study, we develop a *lbest* version of TV_T-BPSO in order to handle the topology optimization problem. Like the original TV_T-BPSO, the *lbest* TV_T-BPSO defines the flight of particles through their velocity and position updates in the binary search space, in order to find the best solution. At each iterative step, the *lbest* TV_T-BPSO updates the velocity $\mathbf{v}_i$ and the position $\mathbf{x}_i$ according to the following equations:

$$\mathbf{v}_i^{k+1} = \mathbf{v}_i^k + c_1 r_1^k (\mathbf{p}_i^k - \mathbf{x}_i^k) + c_2 r_2^k (\mathbf{p}_{g,i}^k - \mathbf{x}_i^k), \tag{5.6}$$

and

$$\mathbf{x}_i^{k+1} = \begin{cases} 0 & \text{if } rand() > TV_T(\mathbf{v}_i^{k+1}, \varphi), \\ 1 & \text{otherwise,} \end{cases} \tag{5.7}$$

respectively, where $\mathbf{p}_{g,i}$ denotes the local best (*lbest*) of the i-th particle, and other symbols have their usual meaning as in Eq. (3.4).

This study adopts the *lbest* TV_T-BPSO in B-SPSO (which we called the modified B-SPSO) so that it can comprehensively explore the topology search space of the truss design problems. In the following section, this modified B-SPSO will be used as an upper level optimizer (for the topology optimization) of the proposed bilevel niching method.

5.5.2 The Proposed Bilevel Niching Method

The general outline for applying the bilevel niching method to the truss problem is presented in Alg. 9, where the outer loop is used for the upper-level optimization and the inner loop is used for the lower level optimization. For clarity, the detail of this algorithm is provided below, where *U/u* and *L/l* denote the steps of the upper-level optimizer (the modified B-SPSO) and the lower level optimizer (PSO), respectively.

Step U1*-Generate an initial swarm of B-SPSO for the topology solutions*: For a truss ground structure with m members and n nodes, an initial population of N_u particles are randomly generated in the m-dimensional search space. The velocity of i-th particle is denoted by $\mathbf{v}_i^u$ which is drawn from $\mathcal{U}$(-v_{max},v_{max}). The position of i-th particle is denoted by $\mathbf{x}_i^u$. Each element of $\mathbf{x}_i^u$ holds a binary number 1 or 0 which represents the presence (or absence) of a member of a ground structure. For example, consider a 10-member, 6-node ground structure (see Fig. 5.1(a)), the position $\mathbf{x}_i^u$=[1011001101] represents a topological solution of this structure where the elements 1, 3, 4, 7, 8, and 10 are present and the element 2, 5, 6, and 9 are absent from the structure. Each particle in the initial population

Algorithm 9 BiL-NM for bilevel truss problems.

```
Initialize the topology (upper level) solutions
repeat
    Check the validity of each topology solution
    for each valid topology do
        Initialize the size (lower level) solutions
        Optimize the size solutions
        Return the optimal size solution
    end for
    Apply niching to all the validate topology solutions
until termination criteria not met
```

represents an initial truss topology based on the random bit information contained in this position vector $\mathbf{x}_i^u$.

Step U2*-Check the validity of each topology solution* $\boldsymbol{x}_i^u$: If the truss topology corresponding to $\mathbf{x}_i^u$ violates constraints G_1 and G_2, then this solution will be ignored by assigning a large penalty as the fitness of $\mathbf{x}_i^u$. Otherwise, this topology will be sent to the lower level optimizer for size optimization.

For the lower level optimization, the standard PSO is used (see section 2.3.2) whose working steps in terms of the truss size optimization are described below.

Step L1*-Generate an initial swarm of PSO for the size solutions of the received* $\boldsymbol{x}_i^u$ *topology*: For this given truss topology $\mathbf{x}_i^u$, an initial population of N_l particles is generated randomly first.

Step L2*-Optimize the size solutions*: Size optimization begins with the evaluation process of the particles used for the lower level optimization. In this case, the fitness value considered constraint violation from [Deb and Gulati 2001] are adopted in this study:

$$f^j(A,\xi) = \begin{cases} 10^7, & \text{if } g_1 \text{ is violated,} \\ C(\mathbf{A},\xi), & \text{otherwise,} \end{cases} \tag{5.8}$$

where $C(\mathbf{A},\xi)=w^j(\mathbf{A},\xi)+10^5\sum_{p=1}^{m}|\langle g_2^p\rangle|+10^5\sum_{q=1}^{n}|\langle g_3^q\rangle|$. Here, $f^j(\mathbf{A},\xi)$ and $w^j(\mathbf{A},\xi)$ denote the fitness value and weight of the truss structure corresponding to the j-th particle. The operator $\langle\rangle$ is the bracket-operator penalty term [Deb and Gulati 2001]. Note that since the standard PSO allows the size variables and shape variables to be bounded within specified limits $[A_{min}\ ,\ A_{max}]$ and $[\xi_{min}\ ,\ \xi_{max}]$, the constraints g_4 and g_5 are automatically satisfied.

After calculating the fitness value of j-th particle, PSO determines the personal best position $\mathbf{p}_j^l$ of this particle and the best global position $\mathbf{p}_g^l$ for the whole swarm. Subsequently, PSO updates $\mathbf{v}_j^l$ and $\mathbf{x}_j^l$ of j-th particle according to Eq. (2.5) and Eq. (2.7), respectively. The lower level optimizer, i.e., the PSO then checks its termination condition. It will terminate the run when a predefined number of iterations is reached, otherwise it

will follow the whole procedure in ***Step L2***.

Step L3*-Returning the optimal size solution for the given truss topology* $\boldsymbol{x}_i^u$: The best size solution that is held by $\mathbf{p}_g^l$ is sent back to the upper level for optimization, as follows.

Step U3*- Apply niching to the topology solutions*: This step involves the following:

1. Sort all the particles: All the particles of modified B-SPSO are sorted in an ascending order according to the fitness values of their personal best positions. The modified B-SPSO stores these sorted particles in a list called P_{sorted}.

2. Determine the species seeds: In this step, the modified B-SPSO uses the particles in P_{sorted} to determine the species based on the niche radius r_s, as described in Chapter 2. Since the upper-level optimization works in a binary-valued space, the species are determined by comparing the Hamming distances between the particles in P_{sorted}. Note that if the topology solution x_i of i-th particle and x_j of j-th particle have the same member connectivity (i.e., the hamming distance between x_i and x_j is zero) as well as their weights are the same, then they will belong to the same species, otherwise they will belong to the different species (see Algorithm 4 in Chapter 2). Here, the species seeds represent the different solutions of a given ground structure. In the end, this algorithm will return S, a list containing all dominating particles, i.e., the species seeds from all identified species, which make up the entire population.

3. Update each *lbest*: Assign $\mathbf{p}_{g,i}^u = \mathbf{x}_i^{seed}$, where $\mathbf{x}_i^{seed}$ is the i-th seed of the species seed set S.

4. Update each $\mathbf{v}_i^u$ and $\mathbf{x}_i^u$: The modified B-SPSO updates its velocity and position according to Eq. (5.6) and Eq. (5.7), respectively.

Step U4*-Stopping criteria*: The upper-level optimizer, i.e., the modified B-SPSO terminates the run when the results do not improve for the predefined number of iterations or function evaluations, otherwise it goes back to ***Step U2***.

5.6 Numerical Examples

Four well-known truss design problems are considered in this section to demonstrate the effectiveness of the proposed bilevel formulation and bilevel niching method (BiL-NM). These problems include the 15-member, 17-member, 39-member [Deb and Gulati 2001, Li et al. 2007, Fenton et al. 2016, Li 2015, Miguel et al. 2013, Luh and Lin 2011, Wu and Tseng 2010], and 72-member [Ho-Huu et al. 2016, Kaveh et al. 2015] ground structures, respectively. For these design problems, the bilevel niching method parameters were set according to the values presented in Table 5.1. Note that the value of the parameters (except r_s) of B-SPSO have been chosen from [Islam et al. 2017]. The value of c_1, c_2, w

Table 5.1: Parameters used in modified B-SPSO (upper level) and PSO (lower level) of the proposed niching method. Here, A_{max} and A_{min} represent the maximum and minimum allowed cross-sectional areas of the members of a given truss problem.

Parameters	Modified B-SPSO	PSO
No. of particles	60	30
Acceleration constant c_1 and c_2	2.00 and 2.00	1.49445 and 1.49445
Inertia weight (w)	1	0.9 to 0.2 (decreasing)
Velocity limit [$-v_{max}$,v_{max}]	[-6,6]	$[A_{min}, A_{max}]$
Position limit [x_{min},x_{max}]	{0,1}	$[0, A_{max}]$
Control parameter limit [φ_{min},φ_{max}]	[1,5]	-
Niche radius r_s	0.0	-

of PSO have been chosen from [Shi and Eberhart 1998]. The velocity and position limits of PSO have been set according to the literature [Deb and Gulati 2001, Li 2015, Luh and Lin 2011].

The results of the proposed BiL-NM are compared with the state-of-the-art methods including GA [Deb and Gulati 2001], FS$_h$-AA) [Luh and Lin 2008], and SCGA [Li 2015]. For all the experiments, the statistical results (including the best, worst, and mean weights, and the standard deviation) are provided for the proposed method over 30 independent runs. It should be noted that the statistical results are calculated based on the best weight found in each run. All the other results are taken from a run that obtains the best results among all the other runs. We allow B-SPSO and PSO to run until it is unable to improve the quality of the solution for 10 consecutive iterations. Note that in all our experiments we consider only topology and size optimization, leaving out the shape optimization for future studies

5.6.1 Selecting the Best Population Sizes (N_u and N_l) and Niche Radius (r_s)

Population size is one of the important parameters of any niching method. It has a significant impact on the performance of such an algorithm. For example, a large population size increases the number of function evaluations, thus experiencing a high computational burden. On the other hand, a small population size limits the niching methods in terms of finding a reasonable amount of good quality solutions. To reduce the computational burden and locating a reasonable amount of design solutions, this study chooses a moderate population size (N_u=60) for the B-SPSO and a standard population size (N_l=30) for the PSO. These two population sizes seem to be sufficient for finding multiple solutions in terms of topology and size of the members of the mentioned truss design problems.

In this study, the niche radius r_s is only the user-defined parameter of BiL-NM which needs to be chosen carefully. In practice, a small value for r_s is chosen for locating multiple

optimal/near optimal solutions of truss design problems. For example, Luh and Lin [Luh and Lin 2008] set the value 0.8 for the niche radius r_s. Likewise, a small value for r_s (=0.3) is also chosen in [Li 2015] to locate the multiple optimal/near optimal solutions for the truss design problems. Considering these, we have conducted several experiments using BiL-NM with different niche radius values (0.1 to 2.5) on the mentioned truss design problems. To do this experiment, the value for the parameters of BiL-NM have been set according to Table 5.1. After the preliminary study, the value for the niche radius r_s of BiL-NM is set to 0.1 which is found to be a better niche radius for the truss design problems of this study.

5.6.2 Example 1: 15-member, 6-node Truss

The 15-member, 6-node ground structure is presented in Fig. 5.4. The material density and modulus of elasticity are 0.1 lb/in.^{3} and 10,000 ksi, respectively. The allowable stress and displacement are ±25 ksi and ±2.0 in., respectively. There are 15 members in this truss and the critical area for these members is set to ϵ=0.09 in.^{2}. The minimal and maximal cross-sectional areas are set to A_{min}=0.00 and A_{max}=35.00 in^2, respectively. To optimize this truss, a vertical load 100,000 lb is considered at node 2 and 3, respectively.

Best Found Solutions

Fig. 5.5 shows the four different topologies obtained by the proposed BiL-NM (from the 15-member, 6-node ground structure). The size solutions of these topologies are provided in Table 5.2. Note that like BiL-NM, the first topology (Fig. 5.5(a)) is also found by the single-stage method: GA [Deb and Gulati 2001] and SCGA [Li 2015], and the two-stage method: FS$_h$-AA [Luh and Lin 2008], respectively. For this topology, BiL-NM found three different size solutions, whereas the compared methods found only a single size solution for the same topology, as shown in Table 5.2. Due to the fact that non-active members are removed at the end of the run and the discrete nature of the upper level search space, duplicate topologies are possible to be present in the upper level population. As a result, it is possible to obtain multiple size solutions for the same topology solution. It can be observed that the weights of the size solutions of BiL-NM are 4707.10, 4712.73, and 4714.36 lb, respectively. On the other hand, the weights of the size solutions of GA, FS$_h$-AA, and SCGA are 4734.34, 4730.82, and 4732.12 lb, respectively. These results show that the weights of the solutions provided by BiL-NM are lighter than the weights of the solutions produced by the compared methods. Like the first topology, the second one (Fig. 5.5(b)) is also found by SCGA [Li 2015]. In this case, the SCGA found a single size solution for this truss topology and its weight is 5011.28 lb, as shown in Table 5.2. For the same topology, the BiL-NM found two different size solutions of the weights 4874.33 and 4949.52 lb, respectively which are 136.95 lb (i.e., 5011.28 - 4874.33) and 61.76 lb (i.e., 5011.28 - 4949.52) lighter than the weights in the solutions found by SCGA [Li 2015]. In

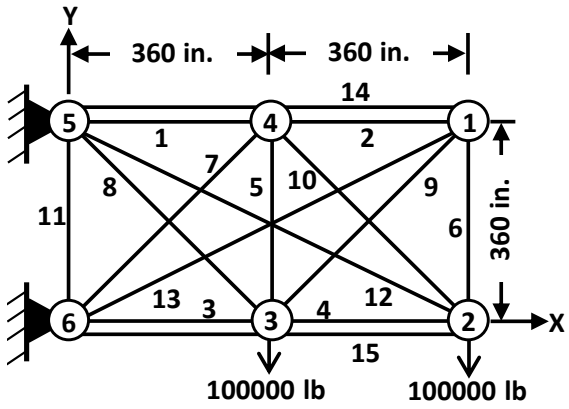


Figure 5.4: Illustration of the 15-member, 6-node ground structure.

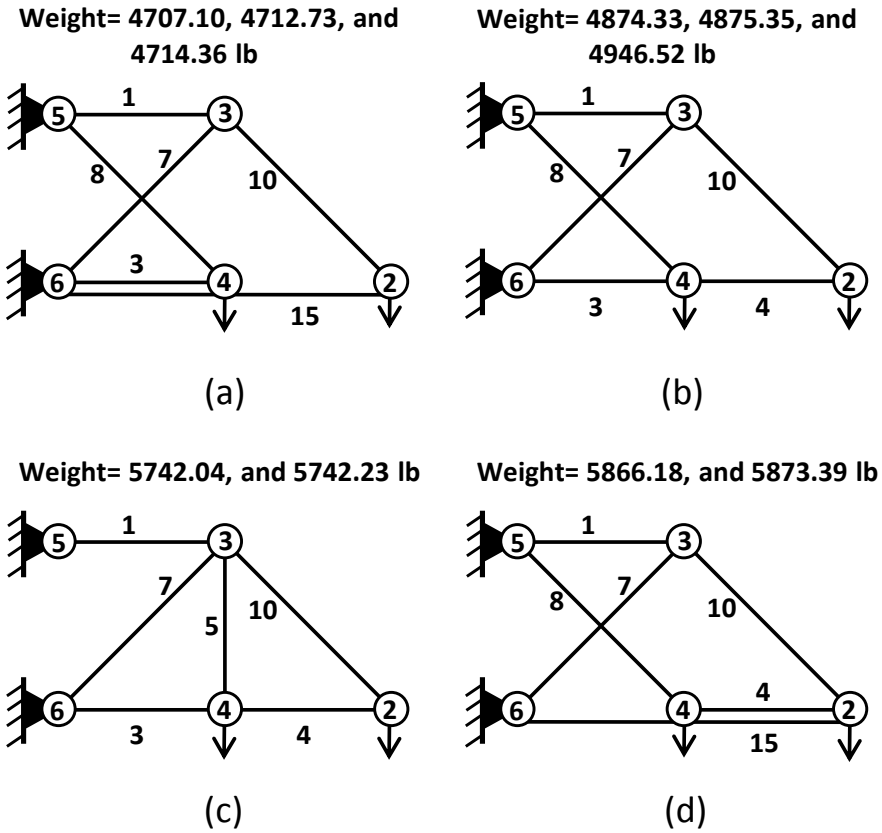


Figure 5.5: Illustration of (a-d) the four different topologies obtained by the proposed BiL-NM, from 15-member, 6-node ground structure.

addition to these topologies, the BiL-NM found two additional topologies for this truss problem, as illustrated in Fig. 5.5(c) and Fig. 5.5(d), respectively. Each of these two topologies consist of two size solutions (see Table 5.2). Since there are no such topologies

Table 5.2: Member areas (in.^{2}) of the design solutions of the 15-member ground structure. Here, the Sol^n.-1, Sol^n.-2, and Sol^n.-3 represent the first, second, and third size solution of a found topology, respectively. The best result is highlighted in bold.

	GA	FS$_h$-AA	SCGA		Proposed BiL-NM								
	Fig. 5.5(a)	Fig. 5.5(a)	Fig. 5.5(a)	Fig. 5.5(b)	Fig. 5.5(a)			Fig. 5.5(b)		Fig. 5.5(c)		Fig. 5.5(d)	
Mem. No.	Sol^n.-1	Sol^n.-1	Sol^n.-1	Sol^n.-1	Sol^n.-1	Sol^n.-2	Sol^n.-3	Sol^n.-1	Sol^n.-2	Sol^n.-1	Sol^n.-2	Sol^n.-1	Sol^n.-2
1	28.187	28.876	28.1577	31.2073	28.4802	29.7820	29.2399	30.0328	29.8143	35.0000	35.0000	29.8772	30.207
3	5.219	5.428	5.2936	21.9711	5.3047	5.5203	5.6702	22.1163	23.1973	17.8406	17.7497		
4				10.3798				15.2584	14.0494	17.7458	18.015	7.7545	6.8447
5										3.9801	4.9944		
7	20.310	20.549	20.2771	25.4616	20.3067	19.1828	19.8427	21.0824	21.1369	34.9996	35.0000	22.3237	22.4403
8	7.772	7.617	7.7142	6.0888	7.6481	7.4989	7.4049	6.045	7.3295			10.6998	11.653
10	20.650	20.265	21.0245	21.9380	20.4701	20.1222	21.4029	20.9488	21.2727	25.058	24.935	21.3113	21.6976
12													
15	14.593	14.3387	14.3387		14.2424	14.7043	13.6203					24.2378	23.5969
Weight (lb)	4731.65	4730.82	4732.12	5011.28	**4707.10**	4712.73	4714.36	4874.33	4946.52	5742.04	5742.23	5808.88	5830.20
σ_{max}(ksi)					19.100	18.858	19.098	23.394	19.294	25.000	25.000	13.217	14609
Mem. no.					8	8	8	8	8	5	5	8	4
δ^{max}(in.)	y-axis				-2.00	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00

for the 15-member, 6-node truss structure as reported in literature, we only provide the weights of the size solutions of these topologies here. The weights of the size solutions of the third topology are 5742.04 and 5742.23 lb, and the weights of the size solutions of the fourth topology are 5866.18 and 5873.39 lb, respectively.

Table 5.2 also presents the maximum yield stresses of the members and displacements of the active nodes of the four topologies obtained by BiL-NM, to show that how close these solutions are to the boundaries of the problem. It can be seen that all the solutions provided by BiL-NM lie on the intersection of some constraints. For example, the yield stress and displacement of the first size solution of the third topology (Fig. 5.5(c)) are 25.00 ksi and -2.00 inches which are equal to the allowable limits. For clarity, we compare results (see Table 5.3) on stresses and nodal displacements for the first size solution of the first topology (Fig. 5.5(a)) of the 15-member truss structure, among several existing optimization methods. Fig. 5.5 and Table 5.2 show that the proposed bilevel niching method is capable of finding multiple topologies and their corresponding size solutions in a single run.

Table 5.4 shows the statistical results of 30 independent runs of BiL-NM for the 15-member, 6-node ground structure. It can be seen that the mean weight and standard deviation are 4708.34 and 0.699, respectively. This shows that the performance of the proposed method is also more reliable on this truss problem.

Efficiency and Sensitivity Analysis

In this section, we compare the efficiency of the proposed BiL-NM with GA [Deb and Gulati 2001] and SCGA [Li 2015], respectively. The reason is that these two methods also performed simultaneous topology and size optimization for this truss problem.

Fig. 5.6 shows the convergence behaviour of BiL-NM when using a fixed population size (30) for the PSO and four different population sizes (20, 40, 60, and 80, respectively)

Table 5.3: Comparisons on stresses and displacements among different methods respectively, for the first topology solution for the 15-member ground structure.

	Stresses in existing members					
	GA	FS$_h$-AA	SCGA	Proposed BiL-NM		
	Fig. 5.5(a)	Fig. 5.5(a)	Fig. 5.5(a)	Fig. 5.5(a)		
Mem. No.	Sol^n-1	Sol^n-1	Sol^n-1	Sol^n-1	Sol^n-2	Sol^n-3
1	7.095	6.92	7.1	7.02	6.71	6.83
3	19.161	-18.42	-18.89	-18.85	-18.11	-17.63
7	-6.962	-6.88	-6.97	-6.96	-7.37	-7.12
8	18.194	18.56	18.33	18.49	18.85	19.09
10	6.865	6.97	6.72	6.90	7.02	6.6
15	-6.852	-6.98	-6.97	-7.02	-6.8	-7.34
Node	Displacement at active nodes					
2	x=-0.69, y=-2.00	x=-0.66, y=-2.00	x=-0.68, y=-1.99	x=-0.51, y=-2.00	x=-0.49, y=-2.00	x=-0.53, y=-2.00
3	x=0.25, y=-0.75	x=0.24, y=-0.74	x=0.25, y=-0.75	x=0.25, y=-0.5	x=0.24, y=-0.77	x=0.25, y=-0.76
4	x=-0.49, y=-2.00	x=-0.50, y=-1.99	x=-0.50, y=-2.0	x=-0.68, y=-2.00	x=-0.65, y=-2.00	x=-0.63, y=-2.00

Table 5.4: Statistical result of 30 independent runs of the BiL-NM for the 15-member ground structure.

Best weight (lb)	Worst weight(lb)	Mean weight (lb)±SD
4707.10	4709.97	4708.34±0.699

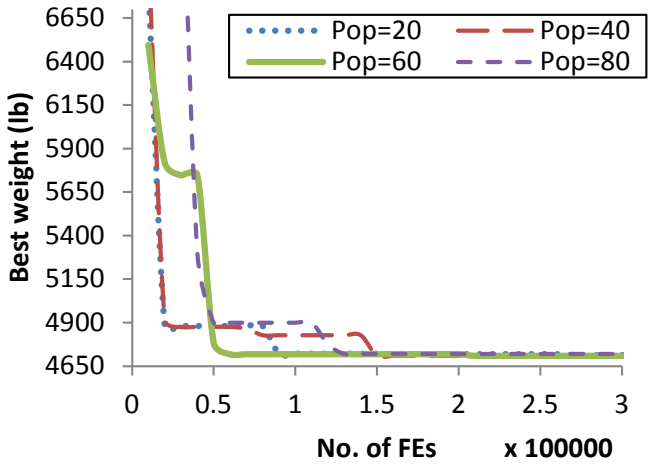


Figure 5.6: Convergence behaviour of BiL-NM with different population sizes for 15-member, 6-node ground structure.

for B-SPSO to optimize the 15-member, 6-node truss structure. It can be seen that BIL-NM solves this problem more efficiently when the population size is set to 60. In this case, BiL-NM reached a similar best weight as that of GA (4731.65 lb) and SCGA (4732.12 lb) after about 55,000 function evaluations (FEs). However, both GA and SCGA reached the same weight after 85,050 and 250,000 FEs [Deb and Gulati 2001, Li 2015] which are higher than the BiL-NM, respectively.

Fig. 5.6 also shows that BiL-NM is not sensitive to the population sizes on this truss problem. BiL-NM can reliably obtain the best weight 4707.10 lb when using different population sizes, with population size of 60 the fastest in reaching this best weight.

5.6.3 Example 2: 17-member, 9-node Truss

This popular 17-member, 9-node truss structure is illustrated in Fig. 5.7(a). The material density and modulus of elasticity are 0.268 lb/in.^{3} and 30,000 ksi, respectively. The allowable stress and displacement are $\pm$50 ksi and $\pm$2.0 in., respectively. There are 17 members in this truss and the critical area for these members is set to ϵ=0.1 inches^2. The minimal and maximal cross-sectional areas are set to A_{min}=0.00 and A_{max}=20.00 in^2, respectively. To optimize this truss structure, a vertical load of 100 kips at node 9 is considered.

Best Found Solutions

In literature, several meta-heuristic methods including the harmony search algorithm (HSA) [Seok and Woo 2004], firefly algorithm (FA), and [Miguel et al. 2013] have been used to study this 17-member ground structure particularly in terms of size optimization. SEOIGE [Fenton et al. 2016] and GP [Assimi et al. 2017] methods perform the topology and size optimization together for this truss problem. However, those state-of-the-art methods, i.e., GA [Deb and Gulati 2001], FS$_h$-AA) [Luh and Lin 2008], and SCGA [Li 2015] did not include this truss problem. Hence, in this study, we compare our results of BiL-NM only with those of FA, SEOIGE, and GP methods.

Fig. 5.7(c-d) shows two different topologies of 17-member truss structure obtained by BiL-NM. The first found topology (Fig. 5.7(c)) consists of 11 members. For this topology, BiL-NM found two different size solutions, whereas GP, FA, and SEOIGE found a single size solution for the same topology (see Table 5.5). It can be observed that the weights of the size solutions of BiL-NM are 2563.15 lb and 2563.79 lb, respectively. The weights provided by GP, FA, and SEOIGE methods are 2575.44 lb, 2581.94 lb, and 2581.90 lb, respectively. These show that the average weight (2563.47 lb) of the size solutions of BiL-NM is less than the average weight (2576.42 lb) of the size solutions provided by the compared methods. The second found topology (Fig. 5.7(d)) consists of 14 members and BiL-NM found only one size solution for this topology, as shown in Table 5.5. To the best of our knowledge, this is the first time in any study which manages to find this second topology and its size solution for the 17-member ground structure (with a weight 2784.24 lb). Table 5.5 shows the maximum yield stresses and nodal displacements of the size solutions related to these two different topologies. It can be observed that the proposed BiL-NM obtained all the design solutions without violating the given stress and displacement constraints.

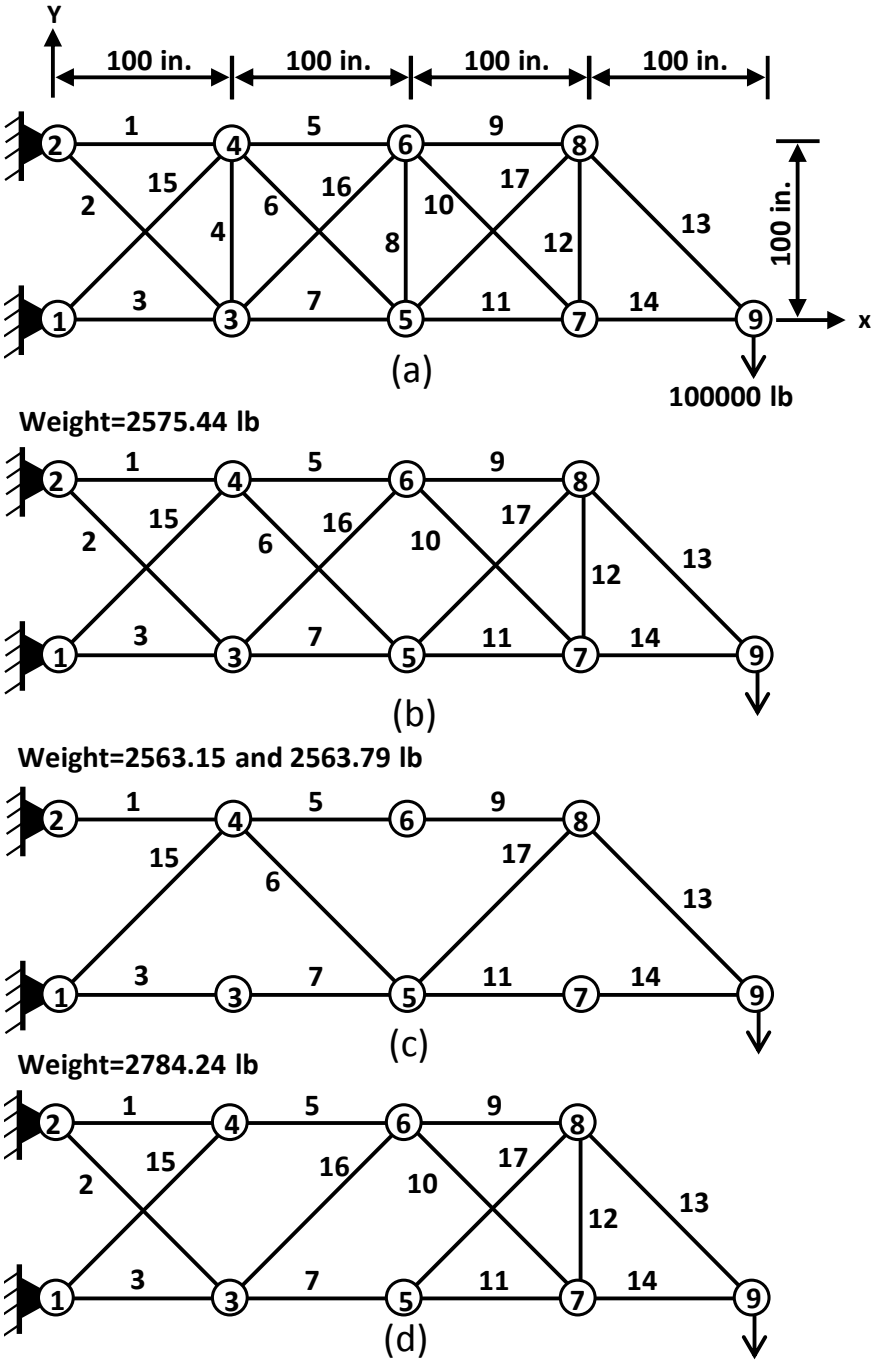


Figure 5.7: Illustration of (a) the 17-member, 9-node ground structure, (b) the design solution employed by GP [Assimi et al. 2017], and (c-d) the design solutions employed by BiL-NM.

Table 5.6 summarizes the results of BiL-NM on the 17-member truss problem. The considerably small standard deviation shows that the performance of BiL-NM is very reliable for this truss problem.

Table 5.5: Member cross-sectional areas (in.^{2}) of the design solutions of 17-member, 9-node ground structure. Here, the Sol^n.-1, and Sol^n.-2 represent the first, and second size solution of a found topology, respectively.

	FA	GP	SEOIGE	Proposed BiL-NM		
	Fig. 5.7(a)	Fig. 5.7(b)	Fig. 5.7(c)	Fig. 5.7(c)		Fig. 5.7(d)
Mem. No.	Sol.^{n}-1	Sol.^{n}-1	Sol.^{n}-1	Sol.^{n}-1	Sol.^{n}-2	Sol.^{n}-1
1	15.896	15.9356		15.0000	15.0000	12.9674
2	0.103	0.1009				5.7125
3	12.092	12.0529		12.0839	11.5911	15.000
4	0.100					
5	8.063	8.0311		8.3796	8.233	12.6931
6	5.590	5.5696		5.5636	5.7828	5.2867
7	11.915	11.9087		11.9534	12.2158	8.3984
8	0.100					0.1429
9	7.965	7.9207		8.237	7.7985	4.1668
10	0.100	0.1011				6.0216
11	4.076	4.0658		4.0031	4.3046	8.5022
12	0.100	0.1011				4.1408
13	5.600	5.6503		5.5604	5.5530	6.0381
14	3.998	4.0159		4.1888	4.0301	4.3076
15	5.548	5.5670		5.8261	5.7767	
16	0.103	0.1009				6.0621
17	5.537	5.5849		5.5288	5.8590	
Weight (lb)	2581.94	2575.44	2581.90	**2563.15**	2563.79	2784.24
σ_{max}(ksi)				26.66	26.66	-26.66
Mem. no.				1	1	3
δ^{max}(in.)				-2.00	-2.00	-2.00

Table 5.6: Statistical result of 30 independent runs of the BiL-NM for the 17-member, 9-node ground structure.

Best weight (lb)	Worst weight(lb)	Mean weight (lb)± SD
2563.15	2567.47	2563.42 ± 0.927

Efficiency and Sensitivity Analysis

Fig. 5.8 shows the convergence behaviour of BiL-NM on the 17-member, 9-node truss problem. BiL-NM provides a faster convergence speed when the population size is set to 60. In this case, it converged to the best weight (2563.15 lb) after only 50,000 FEs. In contrast, FA [Miguel et al. 2013] converged to its best weight (2581.94 lb) after 150,000 FEs. It shows that BiL-NM can provide good quality solutions than FA with fewer number of FEs. Note that the GP [Assimi et al. 2017] and SEOIGE [Fenton et al. 2016] methods

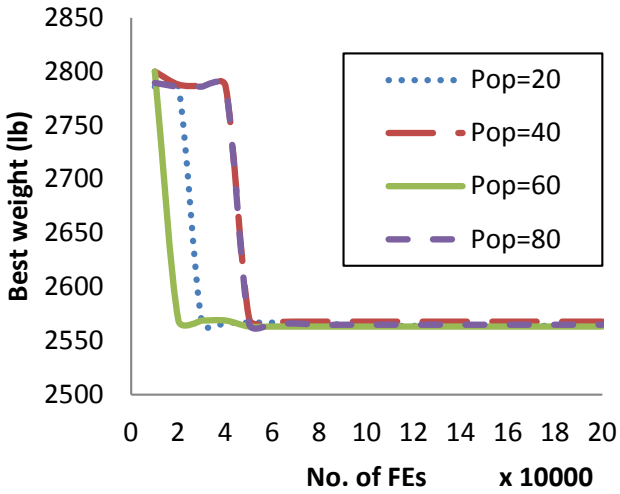


Figure 5.8: Convergence behaviour of BiL-NM with different population sizes for the 17-member, 9-node ground structure.

did not provide the convergence history for this truss problem. Hence, we do not provide a comparison with these algorithms here. Fig. 5.8 also shows that BiL-NM behaved similarly as in Fig. 5.6, i.e., BiL-NM is not sensitive to the different population sizes on this truss problem.

5.6.4 Example 3: 39-member, 10-node Truss

For this experiment we choose a widely-used 3-D truss structure which has an 39-member and 10-node ground structure [Deb and Gulati 2001, Luh and Lin 2008, Li 2015], as shown in Fig. 5.9. Here, members are grouped considering the symmetry on opposite sides, as done by GA [Deb and Gulati 2001] and SCGA [Li 2015]. The members grouping is presented in the followings (see Fig. 5.9)- G1: A_1(1-2); G2: A_2(1-4), A_3(2-3), A_4(1-5), A_5(2-6); G3: A_6(2-4), A_7(2-5), A_8(1-3), A_9(1-6); G4: A_{10}(3-6), A_{11}(4-5), A_{12}(3-4), A_{13}(5-6); G5: A_{14}(3-10), A_{15}(6-7), A_{16}(4-9), A_{17}(5-8); G6: A_{18}(4-7), A_{19}(3-8), A_{20}(5-10), A_{21}(6-9); G7: A_{22}(6-10), A_{23}(3-7), A_{24}(4-8), A_{25}(5-9); G8: A_{26}(5-7), A_{27}(6-8), A_{28}(3-9), A_{29}(4-10); G9: A_{30}(3-5), A_{31}(4-6); G10: A_{32}(1-7), A_{33}(1-10), A_{34}(2-9), A_{35}(2-8); G11: A_{36}(2-7), A_{37}(2-10), A_{38}(1-8), A_{39}(1-9).For this truss structure, the material properties and design parameters are set as follows: Young's modulus and density of the materials are the same as before. The allowable stress and displacement are set to ± 40 ksi and ± 0.35 inches, respectively. The lower (A_{min}) and upper (A_{max}) limits on the member areas are set to be between 0.0 and 35.0 in^2. This truss problem is optimized by applying four forces: (1,000; 1,0000; -5,000) on node 1, (0; 10,000; -5,000) on node 2, and (500; 0; 0) on node 3 and 6, respectively. The same loading condition is also considered by GA [Deb and Gulati 2001], Fs$_h$-AA [Luh and Lin 2008], and SCGA [Li 2015], among which,

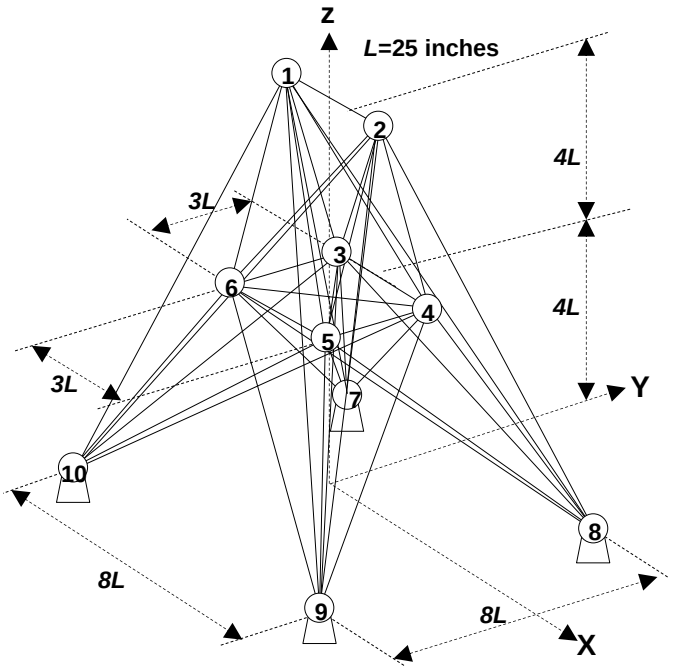


Figure 5.9: Illustration of the 39-member, 10-node ground structure.

SCGA only provides the results of this truss associated with this given loading condition. Therefore, this study only compares the results of BiL-NM with those of SCGA.

Best Found Solutions

For 39-member ground structure, BiL-NM found 4 different topologies which are presented in Fig. 5.10. For these topologies, BiL-NM found different numbers of size solutions for each topology, as shown in Table 5.7. Particularly, BiL-NM found three different size solutions for the first topology, two different size solutions for the second and third topologies, and one size solution for the fourth topology. The cross-sectional areas of the active members of these design solutions as well as their weights are also provided in Table 5.7. It can be observed that the best weight obtained by BiL-NM is 169.90 lb belonging to the first obtained topology. In contrast, the best weight provided by Li [Li 2015] is 224.66 lb, which is 54.76 lb larger. The first obtained topology (Fig. 5.10(a)) has two other size solutions and their weights are 171.51 and 173.37 lb, respectively. The weights of the solutions of the second, third, and fourth topologies (Fig. 5.10(b-d)) range from 171.28 to 174.10 lb. These weights are significantly smaller than the weights found by SCGA. Table 5.7 shows the maximum yield stresses and displacements of the design solutions of BiL-NM. It can be seen that all the design solutions exist in the intersections of some constraints. For example, the yield stress and nodal displacement of the third solution of the first topology are 39.954 ksi and 0.35 in., which are almost the same as

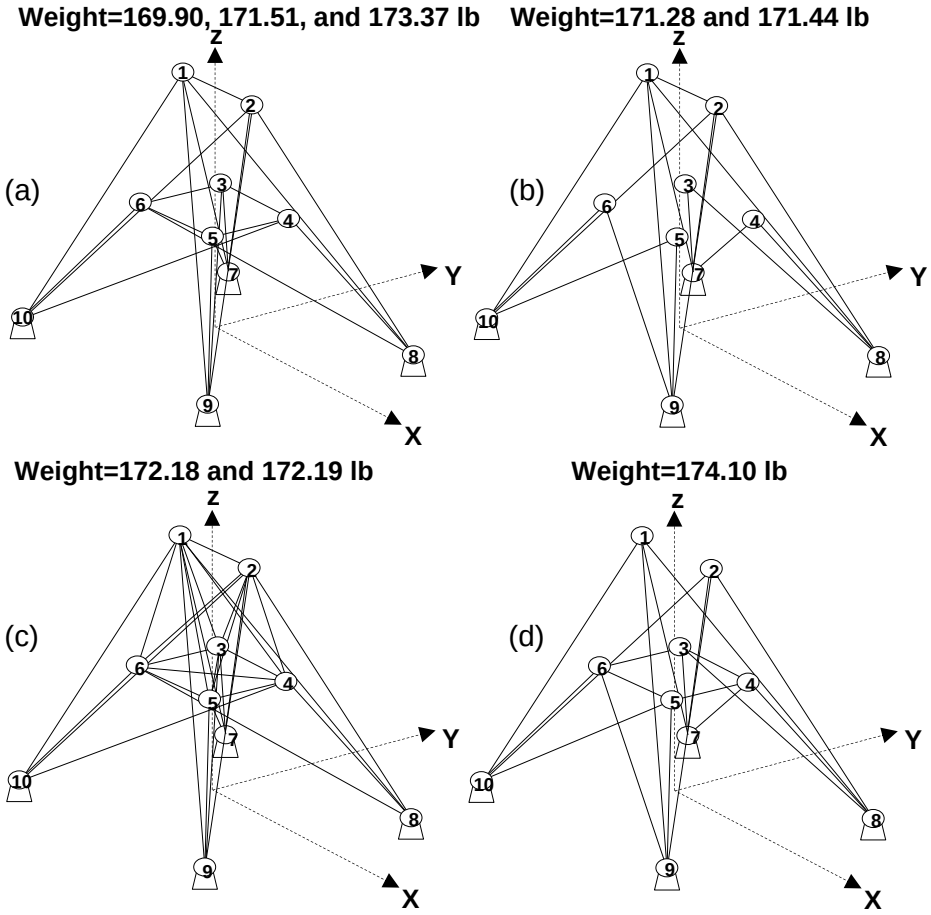


Figure 5.10: Illustration of (a-d) the four different topologies of 39-member, 10-node ground structure employed by BiL-NM.

the allowable limits 40 ksi and 0.35 inches, respectively. Table 5.8 shows again that the mean and standard deviation are 170.15±0.0494 which are considerably good for this truss problem.

Efficiency and Sensitivity Analysis

As shown in Fig. 5.11, like in the previous two scenarios, BiL-NM quickly converged to the best weight with a population size of 60. BiL-NM managed to reach the weight 180 lb only after 10,000 FEs, and 169.90 lb after 170,000 FEs. In contrast, SCGA obtained its best weight which is 224.66 lb after 50,000 FEs [Li 2015]. If we consider the weight 180 lb is the cut-off point, then we would find that the BiL-NM is 5 times faster than the SCGA for this truss problem.

Fig. 5.11 shows that on this truss problem BiL-NM was somewhat slightly sensitive to the population sizes. It is noticeable that BiL-NM again performed the best when population size is set to 60.

Table 5.7: Member cross-sectional areas (in.^{2}) of the design solutions of 39-member, 10-node ground structure. Here, the Sol^n.-1, Sol^n.-2, and Sol^n.-3 represent the first, second, and third size solution of a found topology, respectively.

			Proposed BiL-NM							
	SCGA		Fig. 5.10(a)			Fig. 5.10(b)		Fig. 5.10(c)		Fig. 5.10(d)
Ele. Group	Sol^n.-1	Sol^n.-2	Sol^n.-1	Sol^n.-2	Sol^n.-3	Sol^n.-1	Sol^n.-2	Sol^n.-1	Sol^n.-2	Sol^n.-1
G1	0.55227	0.48337	0.0825	0.0612	0.0315	0.0867	0.0438	0.0432	0.0533	
G2	0.07919	0.00684						0.0066	0.0090	
G3		0.02029						0.0065	0.0093	
G4			0.0166	0.021	0.0237	0.026	0.0310	0.0117		
G5	0.04090	0.04069								
G6						0.0303	0.0309			0.0197
G7		0.09024	0.0091	0.063	0.0193	0.0279	0.0267	0.0172	0.0286	0.0198
G8	0.01154	0.01176	0.011	0.0252	0.0255			0.0333	0.0293	0.0199
G9										0.0200
G10	1.00820	1.01622	1.7499	1.7439	1.7123	1.7499	1.7415	1.7359	1.7382	1.672
G11	1.08541	1.17183	0.0459	0.0513	0.0966	0.0422	0.0544	0.0515	0.0426	0.1548
Weight (lb)	224.66	226.24	**169.90**	171.51	173.37	171.28	171.44	172.18	172.19	174.10
σ_{max}(ksi)			-21.931	-33.333	-39.954	-21.162	-37.177	-36.121	-30.837	-15.967
Mem. no.			1	13	1	1	1	1	1	38
δ^{max}(in.)			0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35

Table 5.8: Statistical result of 30 independent runs of the BiL-NM for the 39-member, 10-node ground structure.

Best weight (lb)	Worst weight(lb)	Mean weight (lb)±SD
169.90	170.19	170.15±0.0494

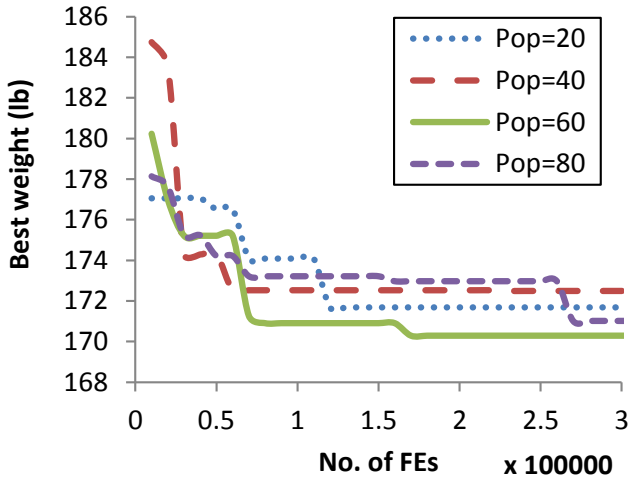


Figure 5.11: Convergence behaviour of BiL-NM with different population sizes for the 39-member, 10-node ground structure.

5.6.5 Example 4: 72-member, 20-node Truss

In this section, we evaluate BiL-NM on a high-dimensional truss problem namely 72-member, 20-node ground structure, as shown in Fig. 5.12. This structure was previously examined by CSP [Kaveh et al. 2014], ADE [Ho-Huu et al. 2016], and FPA [Bekdas et al. 2015] in terms of size optimization. However, in this study we consider to optimize both the topology and size for this high-dimensional truss structure. To the best of our knowledge, very limited studies have been considered to optimize the topology and size for this high-dimensional truss structure e.g., IFA [Wu et al. 2017]. The following settings are considered (according to [Kaveh et al. 2014, Ho-Huu et al. 2016, Bekdas et al. 2015, Wu et al. 2017]): the material density is 0.1 lb/in.^{3}, modulus of elasticity is 10^4 ksi, stress limit is ±25,000 psi, and nodal displacements limit is ±0.25 inches, respectively. The lower (A_{min}) and upper (A_{max}) limits on the member areas are set to be between 0.0 and 03.00 in^2. The structure has 72 members which are divided into 16 groups as follows: G1: A_1-A_4, G2: A_5-A_{12}, G3: A_{13}-A_{16}, G4: A_{17}-A_{18}, G5: A_{19}-A_{22}, G6: A_{23}-A_{30}, G7: A_{31}-A_{34}, G8: A_{35}-A_{36}, G9: A_{37}-A_{40}, G10: A_{41}-A_{48}, G11: A_{49}-A_{52}, G12: A_{53}-A_{54}, G13: A_{55}-A_{58}, G14: A_{59}-A_{66}, G15: A_{67}-A_{70}, and G16: A_{71}-A_{72}. This structure is optimized by applying the load (P_x=5.0 kip, P_y=5.0 kip, and P_z=-5.0 kip) at node 17.

Best Found Solutions

Fig. 5.13 shows two different topology obtained by BiL-NM, from the 72-member truss structure. The first topology (Fig. 5.13(a)) has only 56 members and the second topology (Fig. 5.13(b)) has 54 members, respectively. BiL-NM found two different size solutions for the first topology and three size solutions for the second topology. The cross-sectional areas along with the maximum yield stresses and displacements of these solutions are presented in Table 5.9. The found solutions are compared with those that are reported in literature [Kaveh et al. 2014, Ho-Huu et al. 2016, Bekdas et al. 2015]. It can be observed that the weights of the design solutions employed by the proposed method range from 349.79 lb to 353.74 lb. However, the weights of the design solutions provided by the compared methods range from 379.97 lb to 389.334 lb, which are significantly higher than the weights of the solutions obtained by BiL-NM. Note that the second topology which is presented in Fig. 5.13(b) for the 72-members ground structure is found by the both BiL-NM and improved firefly algorithm (IFA) [Wu et al. 2017]. However, the weight (368.26 lb) found by the IFA method for this topology is higher than the weight (353.70) found the proposed BiL-NM. In this study, the best weight corresponds to the topology in Fig. 5.13(a) which is 349.79 lb. A typical convergence trajectory of the best topology is illustrated in Fig. 5.14. The best result over 30 independent runs has a weight 349.79 lb (see Table 5.9), which is considerably lower than the weights of the solutions in the cited work mentioned in Table 5.9. The results in Table 5.10 again show that that

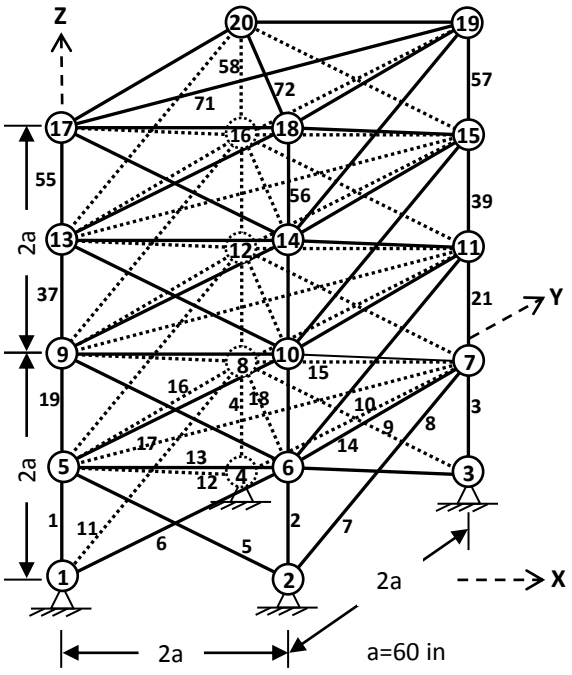


Figure 5.12: Illustration of the 72-member, 20-node ground structure.

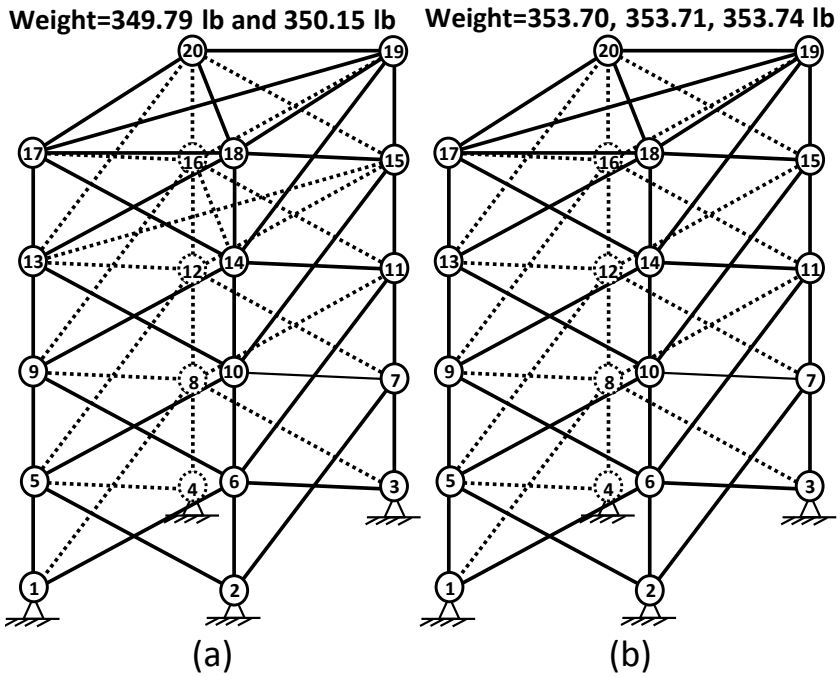


Figure 5.13: Illustration of (a) the first topology, and (b) the second topology of the 72-member, 20-node ground structure obtained by BiL-NM.

6

Table 5.9: Member areas (in.^{2}) of the design solutions of the 72-member, 20-node ground structure. Here, the Sol^n.-1, and Sol^n.-2 represent the first, and second size solution of a found topology, respectively. The best result is highlighted by bold.

	IFA	CSP	FPA	ADE	Proposed BiL-NM				
	Fig. 5.13(b)	Fig. 5.12	Fig. 5.12	Fig. 5.12	Fig. 5.13(a)		Fig. 5.13(b)		
Group	Sol^n-1	Sol^n-1	Sol^n-1	Sol^n-1	Sol^n-1	Sol^n-2	Sol^n-1	Sol^n-2	Sol^n-3
G1	1.989	1.9446	1.8758	1.990	1.8000	1.8000	1.8000	1.8000	1.7997
G2	0.562	0.5026	0.5160	0.563	0.5170	0.4985	0.5396	0.5352	0.5133
G3		0.1000	0.1000	0.111					
G4		0.1000	0.1000	0.111					
G5	1.227	1.2676	1.2993	1.228	1.2228	1.2959	1.2416	1.2573	1.3006
G6	0.442	0.5099	0.5246	0.442	0.4975	0.4977	0.5181	0.5148	0.5266
G7		0.1000	0.1001	0.111					
G8		0.1000	0.1000	0.111					
G9	0.562	0.5067	0.4997	0.563	0.4871	0.5135	0.4991	0.4814	0.4991
G10	0.562	0.5165	0.5089	0.563	0.4945	0.4955	0.5159	0.5138	0.5169
G11		0.1075	0.1000	0.111					
G12	0.111	0.1000	0.1000	0.111	0.1577	0.1237			
G13	0.195	0.1562	0.1575	0.196	0.1027	0.1063	0.1213	0.1213	0.1215
G14	0.562	0.5402	0.5329	0.563	0.5514	0.5353	0.514	0.527	0.5093
G15	0.442	0.4223	0.4089	0.391	0.4072	0.4105	0.4304	0.4468	0.4146
G16	0.562	0.5794	0.5731	0.563	0.5023	0.5747	0.6212	0.5898	0.6711
Weight (lb)	368.26	379.97	379.09	389.334	**349.79**	350.15	353.70	353.71	353.74
σ_{max}(ksi)			-25.00	-25.00	-25.00	-25.00	-25.00	-25.00	-25.00
Mem. no.			55	55	55	55	55	55	55
δ^{max}(in.)			0.25	0.25	0.25	0.25	0.25	0.25	0.25

Table 5.10: Statistical result of 30 independent runs of the BiL-NM for the 72-member, 20-node ground structure.

Best weight (lb)	Worst weight(lb)	Mean weight (lb)± SD
349.79	379.35	355.116±5.449

BiL-NM is a reliable method capable of providing better quality solutions, even on this high-dimensional truss problem.

Efficiency and Sensitivity analysis

Fig. 5.14 shows the convergence behaviour of BiL-NM for the four different population sizes. Like the previous three examples, BiL-NM converges quickly to the best weight in the case of a population size of 60. With this population size, BiL-NM spent 210,000 FEs in order to obtain the best weight 349.79 lb.

As shown in Fig. 5.14, BiL-NM with different population sizes managed to converge to the best weight or close to the best weight, though with different numbers of FEs. It is also apparent from these results and those from the previous experiments, the population size 60 seems to be the good choice for the proposed BiL-NM to find good quality solutions, for both the low- or high-dimensional truss problems.

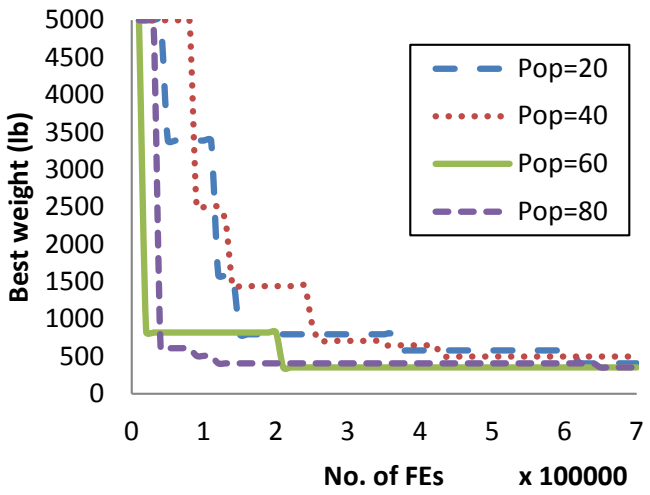


Figure 5.14: Convergence behaviour of BiL-NM with different population sizes for the 72-member, 20-node ground structure.

5.7 Chapter Summary

In this chapter, we have proposed a bilevel formulation for the truss design optimization problem, making it possible to consider optimal solutions in terms of both topology and size without having to treat each as a separable problem. This is usually difficult to achieve with the single- and two-stage approaches of the truss problem as seen in literature. With this bilevel formulation, a bilevel niching method (BiL-NM) is also proposed, where the enhanced B-SPSO has been applied to the upper level optimization while a standard PSO is applied to the lower level. In the enhanced B-SPSO, a time-varying transfer function is employed to enhance its search ability. In this study, the accuracy, stability, and efficiency of the proposed BiL-NM have been examined over the well-known low- and high-dimensional truss design problems. Our numerical studies show that BiL-NM has the ability to find multiple design solutions (topologies) at the upper level of the bilevel truss problem formulation. In addition, it can provide different size solutions for the same topology, i.e., multiple solutions of the member cross-sectional areas at the lower level as well. The results indicate the superiority of the BiL-NM over the state-of-the-art methods on the low-dimensional and as well as high-dimensional truss problems that can be found in literature.

The proposed BiL-NM has the ability to locate multiple topology and size solutions for all types of trusses including roof truss.

CHAPTER 6

Conclusion

In this research, we have been motivated to investigate the relatively unexplored area of structural optimization using techniques from the field of evolutionary computation. Particularly, this research has examined existing evolutionary niching and bilevel techniques and designed an optimization framework for identifying multiple design solutions in terms of topology and size of structural design problems in a single optimization run.

One of the major challenges of obtaining an optimal design solution of a structural design problem is that this design solution should be obtained by means of simultaneous structural topology and size optimization because these two problems are non-separable problems [Deb and Gulati 2001]. Hence, we have formulated the structural optimization problem as a bilevel optimization problem which allows the structural topology optimization in its upper level and at the same time, it allows the size optimization of that topology in its lower level. A bilevel niching method (BiL-NM) has been proposed for finding multiple solutions to a bilevel structural design problem. This study has also proposed a discrete niching method for solving structural topology like multimodal discrete optimization problems. In this thesis, challenging discrete multimodal benchmark instances have been proposed to test the weaknesses and strengths of discrete niching methods. In addition, a time-varying transfer function based binary PSO method has been proposed for the better balance between exploration and exploitation of the search spaces of discrete optimization problems. Finally, the proposed BiL-NM has been applied to a widely known real-world structural optimization problem namely truss optimization problem. The results have shown that the proposed BiL-NM has the ability to find multiple truss topologies and their corresponding size solutions in a single optimization run.

In the following sections, we will revisit the research objectives that have been presented in the introduction chapter. Along with this, the major findings of this study will be presented. Finally, several directions for future research will be discussed, and the concluding remark of the thesis will be presented.

6.1 Research Objectives Revisited

1. To propose guidelines for designing and implementing multimodal discrete benchmark problems and to assess the robustness of a developed discrete niching method on these new benchmark problems.

Multimodal test suites are important for investigating the weaknesses and strengths of any developed niching method. Several attempts have been made to design such test suites specifically for generating continuous multimodal test problems [Mahfoud 1995, Li et al. 2002, Rönkkönen et al. 2008, Li et al. 2013]. Literature shows that there does not exist such multimodal test problems specially designed for discrete optimization problems. To address this issue, Chapter 3 proposed a framework for generating a diverse set of challenging multimodal test instances for well-known discrete optimization problem namely the 0-1 knapsack problem. The framework is flexible allowing a user to have a better control on the settings of the multimodal 0-1 knapsack test instances such as dimensionality, distribution of the optima, the number of optima, and fitness of the optima.

In addition to multimodal benchmarks, Chapter 3 also proposed a binary particle swarm optimization based discrete niching method, specifically a binary speciation-based PSO (B-SPSO). The performance of B-SPSO has been assessed in terms of the *peak ratio*, *success rate*, and *an average number of function evaluations* over the proposed discrete multimodal benchmark problems. The experimental results have shown that the proposed B-SPSO performs well on low-dimensional multimodal 0-1 knapsack instances but often performs poorly as the number of dimensions and optima increase. This suggests that the search performance of B-SPSO need to be improved for the high-dimensional 0-1 multimodal knapsack instances.

2. To identify the limitations of existing discrete optimization methods, specifically, binary PSO and its popular variants in terms of providing optimal/near optimal solution of the real-world optimization problems, and to design a new binary PSO method that overcomes these limitations by means of providing a good balance between exploration and exploitation.

The experimental results in Chapter 3 showed that the proposed B-SPSO performed poorly with the increasing number of dimensions of the proposed multimodal 0-1 knapsack instances. For this research, it is necessary to enhance the searchability of B-SPSO, because it is one of the key optimizers of the proposed bilevel niching method (see Chapter 5).

In Chapter 3, we use the original BPSO for the implementation purpose of the proposed B-SPSO. As the part of the enhancement of B-SPSO, we analyze the exploration and exploitation ability of the original BPSO and its well-known variants. The results have shown that the BPSO and its variants are unable to provide a better exploration of

the search space. In addition, these BPSO's are also unable to provide a good balance between exploration and exploitation. Chapter 4 introduced a modified BPSO namely time-varying transfer function based BPSO (TV_T-BPSO), which adopts a time-varying transfer function to address the shortcoming of existing BPSO's by providing a better balance between exploration and exploitation for the BPSO during its optimization run. To understand the search behaviour of the proposed TV_T-BPSO, a systematic analysis of its exploration and exploitation capability is provided.

The experimental results have shown that TV_T-BPSO outperforms existing BPSO variants on both low-dimensional ($D \leq 500$) and high-dimensional ($1000 \geq D \leq 5000$) classical 0-1 knapsack problems, suggesting that TV_T-BPSO is able to better scale to high-dimensional combinatorial problems than the existing BPSO variants.

3. To formulate the truss design as a bilevel problem, and to design a bilevel niching method that optimizes the bilevel truss problem to find multiple truss topology and size solutions.

Truss design is a well-known structural optimization problem which has important practical applications in various fields. Truss design problems are typically multimodal by nature, meaning that it offers multiple optimal solutions with respect to the topology and/or sizes of the members, but they are evaluated to have similar or equally good objective function values.

Chapter 5 addressed the shortcoming of the existing research works on truss optimization and achieved the above goal by proposing the truss design problems as a bilevel problem. The bilevel formulation allows the truss topology and size optimizations in its two different levels namely the upper level and lower level, respectively. In the upper level, the truss topology is treated as the design variable of the given design problem. In the lower level, the cross-sectional areas of the members of the found topology (in the upper level) are optimized as the part of the size optimization. The proposed formulation allows the truss topology and size optimization working together simultaneously. We proposed a bilevel niching method which consists of two optimizers; the first one is for the topology optimization and the second one is for the size optimization. The bilevel niching method used the modified B-SPSO (see Chapter 5) as the topology optimizer and a standard PSO [Shi and Eberhart 1998] as the size optimizer.

The proposed bilevel niching method is unique than the state-of-the-art methods in three different aspects. Firstly, it has the ability to perform the simultaneous topology and size optimization for the low- to high-dimensional truss design problems. Secondly, it can find multiple design solutions in terms of topology, at the same time it can find multiple size solutions the found topologies in a single optimization run. Finally, it can find multiple design solutions with the less number of function evaluations.

4. To demonstrate the accuracy, robustness, and efficiency of the proposed bilevel niching method, compared with other well-known truss optimization methods over various challenging low and high-dimensional truss design problems.

Performance evaluation is one of the important tasks for the validation of a niching method. This should be done by conducting a comparative study between the developed niching method and other state-of-the-art niching methods over a diverse set of real or artificial benchmark problems. Chapter 5 addressed the limitations of existing studies on niching-based truss optimization and achieved the above goal by analyzing the accuracy, robustness, and efficiency of the proposed bilevel niching method over the real-world challenging low- and high-dimensional truss design problems [Deb and Gulati 2001, Luh and Lin 2008, Li 2015, Ho-Huu et al. 2016].

The numerical studies have shown that the proposed bilevel niching method can find multiple design solutions (topologies) at the upper level of the bilevel truss problem formulation. In addition, it can provide different size solutions for the same topology, i.e., multiple solutions of the member cross-sectional areas at the lower level as well. The results indicate the superiority (in terms of the accuracy, robustness, and efficiency) of the proposed bilevel niching method over the state-of-the-art methods [Deb and Gulati 2001, Luh and Lin 2008, Li 2015] on the given low-dimensional and as well as high-dimensional truss design problems.

6.2 Future Research

In this thesis, we took a major step in formulating the use of bilevel approach for finding multiple topology and size solutions for challenging structural design problems. We have also provided a practical framework for solving such design problems. However, there remain several important aspects that require further investigations.

Multimodal Benchmark Suite

The focus of this thesis was mainly on multimodal structural design problems. For this, we have developed a bilevel niching method using a binary SPSO (B-SPSO) niching method and a standard PSO. The B-SPSO is a newly developed discrete niching method which has to be examined over the discrete multimodal benchmark problems, before using it for the real-world optimization problems. However, there are no well-known discrete multimodal benchmark problems in the literature which can be used for this purpose. Therefore, this study has designed a new benchmark suite for producing the multimodal 0-1 knapsack benchmark problems. With respect to the new benchmark suite, the following areas

require further investigation.

- Designing new types of multimodal 0-1 knapsack instances: The 0-1 knapsack problem has been widely studied in the previous decades. In the recent years, several studies have shown the interest on a new type of instance 0-1 knapsack instance namely the *spanner instance* [Pisinger 2005, Patvardhan et al. 2014; 2015, Agra et al. 2016]. In this study, we have shown only the design procedure for the multimodal *strongly correlated* and *subset sum instances*. So, there is still room for the improvement of the proposed benchmark suite by adopting the design procedure of multimodal spanner instances.

- Discover multiple solutions for other challenging combinatorial optimization problems: Despite the 0-1 knapsack problem, many other real-world problems belong to the family of combinatorial optimization problems and they are also multimodal by nature e.g., the traveling salesman problem [Tsai et al. 2004], vehicle routing problem [Gendreau et al. 1994a], cryptography [Laskari et al. 2005], and design optimization problem [Deb and Gulati 2001]. To facilitate analyzing the multimodal properties of these combinatorial problems, well-defined benchmarks are required. As an initiative, this study proposed a multimodal benchmark suite for producing the multimodal 0-1 knapsack instances. However, there is a need to discover multiple solutions for other challenging real-world multimodal combinatorial problems, from a decision maker's perspective.

Time-varying Transfer Function Based BPSO (TV_T-BPSO)

In the previous section, we briefly explored some potential future research topics in the context of multimodal benchmark instances specifically design for the 0-1 knapsack problem. In this section, we explore several potential research directions with respect to improve the performance of TV_T-BPSO.

- Designing a new velocity update equation: In Chapter 4, we have shown that the BPSO's have the difficulties of providing a good balance between exploration and exploitation during an optimization run. This study identified one of the reasons for this problem is that the static nature of the transfer functions of BPSO's. Hence, a time-varying transfer function has been proposed for the BPSO in Chapter 4. In addition to this, the research on the exploration and exploitation balancing can be extended by developing new velocity update equations for the BPSO.

- Performance test over other challenging real-world discrete optimization problems: In this study, the performance of TV_T-BPSO has been examined over the low and high dimensional 0-1 knapsack benchmark problems. However, this study can be

extended in terms of the performance evaluation over the well-known discrete optimization problems such as the traveling salesman problem [Tsai et al. 2004], and scheduling problem [Gendreau et al. 1994a].

Multimodal Truss Structure Design Framework

In this section, we explore several potential research directions with respect to the enhancement of the proposed bilevel truss optimization framework.

- Consideration of realistic design constraints: In literature, most of the research works on the simultaneous truss topology and size optimization have been considered two well-known constraints namely stress and displacement constraints. However, the design optimization of truss structures with fundamental or multiple-frequency constraints is extremely useful when improving the dynamic performance of structures [Grandhi 1993, Sedaghati et al. 2002]. Due to this, the topology and size optimization of truss structures with frequency constraints is one of the active areas in the research of structural optimization at present [Zuo et al. 2011, Kaveh and Zolghadr 2012, Gomes 2011, Kaveh and Zolghadr 2014, Farshchin et al. 2016a;b]. In this study, the proposed bilevel truss optimization framework only considered the stress and displacement constraints. However, there are still rooms for the improvement of the proposed framework by means of including the more realistic constraints such as the natural frequency constraint.

Bibliography

A. Acan and A. Ünveren. A great deluge and tabu search hybrid with two-stage memory support for quadratic assignment problem. *Applied Soft Computing*, 36:185–203, 2015.

H. Adeli and O. Kamal. Efficient optimization of plane trusses. *Advances in Engineering Software and Workstations*, 13(3):116 – 122, 1991.

A. Agra, C. Requejo, and E. Santos. Implicit cover inequalities. *Journal of Combinatorial Optimization*, 31(3):1111–1129, Apr 2016.

A. Ahrari and K. Deb. An improved fully stressed design evolution strategy for layout optimization of truss structures. *Computers & Structures*, 164:127–144, 2016.

A. Ahrari, A. A. Atai, and K. Deb. Simultaneous topology, shape and size optimization of truss structures by fully stressed design based on evolution strategy. *Engineering Optimization*, 47(8):1063–1084, 2015.

E. Aiyoshi and K. Shimizu. A solution method for the static constrained stackelberg problem via penalty method. *IEEE Transactions on Automatic Control*, 29(12):1111–1114, 1984.

D. Aksen and N. Aras. A bilevel fixed charge location model for facilities under imminent attack. *Computers & Operations Research*, 39(7):1364–1381, 2012.

F. A. Al-Khayyal, R. Horst, and P. M. Pardalos. Global optimization of concave functions subject to quadratic constraints: an application in nonlinear bilevel programming. *Annals of Operations Research*, 34(1):125–147, 1992.

J. Arora. *Introduction to optimum design*. Elsevier, 2004.

L. Asadzadeh. A local search genetic algorithm for the job shop scheduling problem with intelligent agents. *Computers & Industrial Engineering*, 85:376–383, 2015.

H. Assimi, A. Jamali, and N. Nariman-zadeh. Sizing and topology optimization of truss structures using genetic programming. *Swarm and Evolutionary Computation*, pages –, 2017. ISSN 2210 - 6502. doi: https://doi.org/10.1016/j.swevo.2017.05.009.

E. Balas and E. Zemel. An algorithm for large zero-one knapsack problems. *Operations Research*, 28(5):1130–1154, 1980.

R. J. Balling, R. R. Briggs, and K. Gillman. Multiple optimum size/shape/topology designs for skeletal structures using a genetic algorithm. *Journal of structural engineering*, 132(7):1158–1165, 2006.

H. Banka and S. Dara. A hamming distance based binary particle swarm optimization (hdbpso) algorithm for high dimensional feature selection, classification and validation. *Pattern Recognition Letters*, 52:94 – 100, 2015.

J. C. Bansal and K. Deep. A modified binary particle swarm optimization for knapsack problems. *Applied Mathematics and Computation*, 218(22):11042–11061, 2012.

J. F. Bard. *Practical Bilevel Optimization*. Springer US, 1998.

J. F. Bard and J. T. Moore. A branch and bound algorithm for the bilevel programming problem. *SIAM Journal on Scientific and Statistical Computing*, 11(2):281–292, 1990.

D. Beasley, D. R. Bull, and R. R. Martin. A sequential niche technique for multimodal function optimization. *Evolutionary computation*, 1(2):101–125, 1993.

Z. Beheshti, S. M. Shamsuddin, and S. S. Yuhaniz. Binary accelerated particle swarm algorithm (BAPSA) for discrete optimization problems. *Journal of Global Optimization*, 57(2):549–573, 2013.

G. Bekdas, S. M. Nigdeli, and X. she Yang. Sizing optimization of truss structures using flower pollination algorithm. *Applied Soft Computing*, 37:322 – 331, 2015.

J. E. Bell and P. R. McMullen. Ant colony optimization techniques for the vehicle routing problem. *Advanced engineering informatics*, 18(1):41–48, 2004.

M. P. Bendsøe. Optimal shape design as a material distribution problem. *Structural and multidisciplinary optimization*, 1(4):193–202, 1989.

K. K. Bhattacharjee and S. Sarmah. Shuffled frog leaping algorithm and its application to 0/1 knapsack problem. *Applied Soft Computing*, 19(0):252–263, 2014. ISSN 1568-4946.

M. R. Bonyadi and Z. Michalewicz. Particle swarm optimization for single objective continuous space problems: a review, 2017.

N. Bozorgi, M. Abedzadeh, and M. Zeinali. Tabu search heuristic for efficiency of dynamic facility layout problem. *The International Journal of Advanced Manufacturing Technology*, 77(1-4):689–703, 2015.

J. Bracken and J. T. McGill. Mathematical programs with optimization problems in the constraints. *Operations Research*, 21(1):37–44, 1973.

P. Brandimarte. Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations research*, 41(3):157–183, 1993.

R. Brits, A. P. Engelbrecht, and F. V. den Bergh. A niching particle swarm optimizer. In *Proceedings of the 4th Asia-Pacific conference on simulated evolution and learning*, volume 2, pages 692–696. Singapore: Orchid Country Club, 2002.

D. Chang, Y. Zhao, and Y. Xiao. A robust dynamic niching genetic clustering approach for image segmentation. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 1077–1084. ACM, 2011.

W.-N. Chen and J. Zhang. An ant colony optimization approach to a grid workflow scheduling problem with various qos requirements. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 39(1):29–43, 2009.

W.-C. Chiang and P. Kouvelis. An improved tabu search heuristic for solving facility layout design problems. *International Journal of Production Research*, 34(9):2565–2585, 1996.

P. W. Christensen and A. Klarbring. *An Introduction to Structural Optimization*. Springer Netherlands, 2009.

R. Christensen. *Theory of viscoelasticity: an introduction*. Elsevier, 2012.

A. N. Christiansen, J. A. Bærentzen, M. Nobel-Jørgensen, N. Aage, and O. Sigmund. Combined shape and topology optimization of 3d structures. *Computers & Graphics*, 46:25–35, 2015.

S. Christiansen, M. Patriksson, and L. Wynter. Stochastic bilevel programming in structural optimization. *Structural and multidisciplinary optimization*, 21(5):361–371, 2001.

C.-S. Chung, M. S. Hung, and W. O. Rom. A hard knapsack problem. *Naval Research Logistics (NRL)*, 35(1):85–98, 1988.

V. Chvátal. Hard knapsack problems. *Operations Research*, 28(6):1402–1411, 1980.

B. Colson, P. Marcotte, and G. Savard. An overview of bilevel optimization. *Annals of Operations Research*, 153(1):235–256, 2007.

S. Das, S. Maity, B.-Y. Qu, and P. Suganthan. Real-parameter evolutionary multimodal optimization: A survey of the state-of-the-art. *Swarm and Evolutionary Computation*, 1(2):71–88, 2011. ISSN 2210-6502.

K. A. De Jong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* PhD thesis, Ann Arbor, MI, USA, 1975. AAI7609381.

J. D. Deaton and R. V. Grandhi. A survey of structural and multidisciplinary continuum topology optimization: post 2000. *Structural and Multidisciplinary Optimization*, 49 (1):1–38, 2014.

K. Deb. Genetic algorithms in multimodal function optimization, the clearinghouse for genetic algorithms. Master's thesis, University of Alabama, Tuscaloosa, 1989.

K. Deb and S. Gulati. Design of truss-structures for minimum weight using genetic algorithms. *Finite Elements in Analysis and Design*, 37(5):447 – 465, 2001.

S. Degertekin. Harmony search algorithm for optimum design of steel frame structures: a comparative study with other optimization methods. *Structural Engineering and Mechanics*, 29(4):391–410, 2008.

S. Dempe. *Foundations of Bilevel Programming.* Springer US, 2002.

A. Dominguez, I. Stiharu, and R. Sedaghati. Practical design optimization of truss structures using the genetic algorithms. *Research in Engineering Design*, 17(2):73–84, 2006.

M. Dorigo and L. M. Gambardella. Ant colonies for the travelling salesman problem. *biosystems*, 43(2):73–81, 1997.

M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1):29–41, 1996.

M. Dorigo, G. D. Caro, and L. M. Gambardella. Ant algorithms for discrete optimization. *Artificial life*, 5(2):137–172, 1999.

M. Dorigo, D. de Recherches Du Fnrs Marco Dorigo, and T. Stützle. *Ant Colony Optimization.* A Bradford book. Bradford book, 2004. ISBN 9780262042192.

D.-L. Duan, X.-D. Ling, X.-Y. Wu, and B. Zhong. Reconfiguration of distribution network for loss reduction and reliability improvement based on an enhanced genetic algorithm. *International Journal of Electrical Power & Energy Systems*, 64:88–95, 2015.

M. G. Epitropakis, V. P. Plagianakos, and M. N. Vrahatis. Multimodal optimization using niching differential evolution with index-based neighborhoods. In *2012 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2012.

J. B. Escario, J. F. Jimenez, and J. M. Giron-Sierra. Ant colony extended: experiments on the travelling salesman problem. *Expert Systems with Applications*, 42(1):390–410, 2015.

Z. Ezziane. Solving the 0/1 knapsack problem using an adaptive genetic algorithm. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 16(1):23–30, Jan. 2002. ISSN 0890-0604.

M. Farshchin, C. Camp, and M. Maniat. Multi-class teaching–learning-based optimization for truss design with frequency constraints. *Engineering Structures*, 106:355–369, 2016a.

M. Farshchin, C. V. Camp, and M. Maniat. Optimal design of truss structures for size and shape with frequency constraints using a collaborative optimization strategy. *Expert Systems with Applications*, 66:203–218, 2016b.

M. Fenton, C. McNally, J. Byrne, E. Hemberg, J. McDermott, and M. ONeill. Discrete planar truss optimization by node position variation using grammatical evolution. *IEEE Transactions on Evolutionary Computation*, 20(4):577–589, 2016.

J. E. Fieldsend. Using an adaptive collection of local evolutionary algorithms for multimodal problems. *Soft Computing*, 19(6):1445–1460, 2014.

F. Flager, G. Soremekun, A. Adya, K. Shea, J. Haymaker, and M. Fischer. Fully constrained design: A general and scalable method for discrete member sizing optimization of steel truss structures. *Computers and Structures*, 140:55 – 65, 2014.

H. Fredricson, T. Johansen, A. Klarbring, and J. Petersson. Topology optimization of frame structures with flexible joints. *Structural and multidisciplinary optimization*, 25 (3):199–214, 2003.

A. Friedlander and F. A. M. Gomes. Solution of a truss topology bilevel programming problem by means of an inexact restoration method. *Computational and Applied Mathematics*, 30:109 – 125, 2011.

B. S. Gan, T. Hara, A. Han, S. W. Alisjahbana, and S. Asad. Evolutionary aco algorithms for truss optimization problems. *Procedia Engineering*, 171:1100 – 1107, 2017.

M. Gen and R. Cheng. *Genetic algorithms and engineering optimization*, volume 7. John Wiley & Sons, 2000.

M. Gen, F. Altiparmak, and L. Lin. A genetic algorithm for two-stage transportation problem using priority-based encoding. *OR spectrum*, 28(3):337–354, 2006.

M. Gendreau, A. Hertz, and G. Laporte. A tabu search heuristic for the vehicle routing problem. *Management Science*, 40(10):1276–1290, 1994a.

M. Gendreau, A. Hertz, and G. Laporte. A tabu search heuristic for the vehicle routing problem. *Management science*, 40(10):1276–1290, 1994b.

A. Ghosh and A. K. Mallik. *Theory of Mechanisms and Machines*. Affilated East-West Press, NewDelhi, 2011. ISBN 8185938938.

F. Glover. Tabu searchpart i. *ORSA Journal on computing*, 1(3):190–206, 1989.

D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989. ISBN 0201157675.

D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49, Hillsdale, NJ, USA, 1987. L. Erlbaum Associates Inc. ISBN 0-8058-0158-8.

D. E. Goldberg and M. P. Samtani. Engineering optimization via genetic algorithm. In *Electronic computation*, pages 471–482. ASCE, 1986.

H. M. Gomes. Truss optimization with dynamic constraints using a particle swarm algorithm. *Expert Systems with Applications*, 38(1):957–968, 2011.

J. F. Gonçalves, J. J. de Magalhães Mendes, and M. G. Resende. A hybrid genetic algorithm for the job shop scheduling problem. *European journal of operational research*, 167(1):77–95, 2005.

J. Gordon. *Structures or Why things dont fall down*. Springer US, 2012. ISBN 9781461590743. URL https://books.google.com.bd/books?id=2VfTBwAAQBAJ.

R. Grandhi. Structural optimization with frequency constraints- a review. *AIAA Journal*, 31(12):2296–2303, 1993.

C. Gupta. *Optimization Techniques in Operation Research*. IK International Pvt Ltd, 2008.

P. Hajela, E. Lee, and C.-Y. Lin. Genetic algorithms in structural topology optimization. In *Topology design of structures*, pages 117–133. Springer, 1993.

W. Halter and S. Mostaghim. Bilevel optimization of multi-component chemical systems using particle swarm optimization. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 1240–1247. IEEE, 2006.

R. W. Hamming. Error detecting and error correcting codes. *Bell System technical journal*, 29(2):147–160, 1950.

L. Han, C. Huang, S. Zheng, Z. Zhang, and L. Xu. Vanishing point detection and line classification with bpso. *Signal, Image and Video Processing*, 11(1):17–24, 2017.

Y.-C. He, X.-Z. Wang, Y.-L. He, S.-L. Zhao, and W.-B. Li. Exact and approximate algorithms for discounted 0-1 knapsack problem. *Information Sciences*, 369:634 – 647, 2016.

J. Herskovits, A. Leontiev, G. Dias, and G. Santos. Contact shape optimization: a bilevel programming approach. *Structural and multidisciplinary optimization*, 20(3):214–221, 2000.

A. J. Higgins, S. Hajkowicz, and E. Bui. A multi-objective model for environmental investment decision making. *Computers and Operations Research*, 35(1):253–266, 2008. ISSN 0305-0548.

V. Ho-Huu, T. Nguyen-Thoi, T. Vo-Duy, and T. Nguyen-Trang. An adaptive elitist differential evolution for optimization of truss structures with discrete design variables. *Computers and Structures*, 165:59–75, 2016.

J. H. Holland. *Adaptation in natural and artificial Systems*. MIT Press, Cambridge, MA, USA, 1992a. ISBN 0-262-58111-6.

J. H. Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992b.

X. Huang and Y. Xie. Evolutionary topology optimization of continuum structures with an additional displacement constraint. *Structural and Multidisciplinary Optimization*, 40(1):409–416, 2010.

M. J. Islam, X. Li, and Y. Mei. A time-varying transfer function for balancing the exploration and exploitation ability of a binary {PSO}. *Applied Soft Computing*, 59:182 – 196, 2017.

B. Jarboui, N. Damak, P. Siarry, and A. Rebai. A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems. *Applied Mathematics and Computation*, 195(1):299–308, 2008.

Y. W. Jeong, J. B. Park, S. H. Jang, and K. Y. Lee. A new quantum-inspired binary pso: Application to unit commitment problems for power systems. *IEEE Transactions on Power Systems*, 25(3):1486–1495, 2010.

J.-B. Jo, Y. Li, and M. Gen. Nonlinear fixed charge transportation problem by spanning tree-based genetic algorithm. *Computers & Industrial Engineering*, 53(2):290–298, 2007.

A. R. Jordehi and J. Jasni. Particle swarm optimisation for discrete optimisation problems: a review. *Artificial Intelligence Review*, 43(2):243–258, 2015.

V. V. Kalashnikov, R. C. Herrera Maldonado, J.-F. Camacho-Vallejo, and N. I. Kalashnykova. A heuristic algorithm solving bilevel toll optimization problems. *The International Journal of Logistics Management*, 27(1):31–51, 2016.

D. Karaboga. An idea based on honey bee swarm for numerical optimization. Technical report, Technical report-tr06, Erciyes university, engineering faculty, computer engineering department, 2005.

A. Kaveh and S. Talatahari. An improved ant colony optimization for the design of planar steel frames. *Engineering Structures*, 32(3):864–873, 2010.

A. Kaveh and A. Zolghadr. Truss optimization with natural frequency constraints using a hybridized css-bbbc algorithm with trap recognition capability. *Computers & Structures*, 102:14–27, 2012.

A. Kaveh and A. Zolghadr. Democratic pso for truss layout and size optimization with frequency constraints. *Computers & Structures*, 130:10–21, 2014.

A. Kaveh, R. Sheikholeslami, S. Talatahari, and M. Keshvari-Ilkhichi. Chaotic swarming of particles: A new method for size optimization of truss structures. *Advances in Engineering Software*, 67:136 – 147, 2014.

A. Kaveh, B. Mirzaei, and A. Jafarvand. An improved magnetic charged system search for optimization of truss structures with continuous and discrete variables. *Applied Soft Computing*, 28:400 – 410, 2015.

J. Kennedy. Particle swarm optimization. In *Encyclopedia of machine learning*, pages 760–766. Springer, 2011.

J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks, 1995*, volume 4, pages 1942–1948, Nov 1995.

J. Kennedy and R. C. Eberhart. A discrete binary version of the particle swarm algorithm. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, volume 5, pages 4104–4108, Oct 1997.

U. Kirsch. Optimal topologies of truss structures. *Computer Methods in Applied Mechanics and Engineering*, 72(1):15 – 28, 1989.

M. Kočvara. Topology optimization with displacement constraints: a bilevel programming approach. *Structural optimization*, 14(4):256–263, 1997.

J. R. Koza. *Genetic programming: on the programming of computers by means of natural selection*, volume 1. MIT press, 1992.

S. Krenk and J. Høgsberg. *Truss Structures*, pages 39–89. Springer Netherlands, Dordrecht, 2013.

L. Krog, A. Tucker, M. Kemp, and R. Boyd. Topology optimization of aircraft wing box ribs. In *10th AIAA/ISSMO multidisciplinary analysis and optimization conference*, pages 1–11, 2004.

E. Laskari, G. Meletiou, Y. Stamatiou, and M. Vrahatis. Evolutionary computation based cryptanalysis: A first study. *Nonlinear Analysis: Theory, Methods and Applications*, 63 (5):823–830, 2005.

S. Lee, S. Soak, S. Oh, W. Pedrycz, and M. Jeon. Modified binary particle swarm optimization. *Progress in Natural Science*, 18(9):1161–1166, 2008.

J.-P. Li. Truss topology optimization using an improved species-conserving genetic algorithm. *Engineering Optimization*, 47(1):107–128, 2015.

J.-P. Li, M. E. Balazs, G. T. Parks, and P. J. Clarkson. A species conserving genetic algorithm for multimodal function optimization. *Evolutionary computation*, 10(3):207–234, 2002.

L. Li, Z. Huang, F. Liu, and Q. Wu. A heuristic particle swarm optimizer for optimization of pin connected structures. *Computers and Structures*, 85:340–349, 2007.

X. Li. Adaptively choosing neighbourhood bests using species in a particle swarm optimizer for multimodal function optimization. In *Genetic and Evolutionary Computation–GECCO 2004*, pages 105–116. Springer, 2004.

X. Li. Niching without niching parameters: Particle swarm optimization using a ring topology. *IEEE Transactions on Evolutionary Computation*, 14(1):150–169, Feb 2010. ISSN 1089-778X.

X. Li, P. Tian, and X. Min. A hierarchical particle swarm optimization for solving bilevel programming problems. In *International Conference on Artificial Intelligence and Soft Computing*, pages 1169–1178. Springer, 2006.

X. Li, A. Engelbrecht, and M. G. Epitropakis. Benchmark functions for CEC'2013 special session and competition on niching methods for multimodal function optimization. *RMIT University, Evolutionary Computation and Machine Learning Group, Australia, Tech. Rep.*, 2013.

X. Li, M. G. Epitropakis, K. Deb, and A. Engelbrecht. Seeking multiple solutions: An updated survey on niching methods and their applications. *IEEE Transactions on Evolutionary Computation*, 21(4):518–538, 2017.

C.-J. Liao, C.-T. Tseng, and P. Luarn. A discrete version of particle swarm optimization for flowshop scheduling problems. *Computers and Operations Research*, 34(10):3099–3111, 2007.

J. C.-W. Lin, L. Yang, P. Fournier-Viger, T.-P. Hong, and M. Voznak. A binary pso approach to mine high-utility itemsets. *Soft Computing*, pages 1–19, 2016.

D. Liu, Q. Jiang, and J. X. Chen. Binary inheritance learning particle swarm optimisation and its application in thinned antenna array synthesis with the minimum sidelobe level. *IET Microwaves, Antennas Propagation*, 9(13):1386–1391, 2015a.

J. Liu, Y. Mei, and X. Li. An analysis of the inertia weight parameter for binary particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, PP(99):1–1, 2015b.

J.-H. Liu, R.-H. Yang, and S.-H. Sun. The analysis of binary particle swarm optimization. *Journal of Nanjing University (Natural Sciences)*, 47(5):504–514, 2011.

G.-C. Luh and C.-H. Chueh. Multi-modal topological optimization of structure using immune algorithm. *Computer Methods in Applied Mechanics and Engineering*, 193 (36-38):4035–4055, 2004. ISSN 0045-7825.

G.-C. Luh and C.-Y. Lin. Optimal design of truss structures using ant algorithm. *Structural and Multidisciplinary Optimization*, 36(4):365–379, 2008.

G.-C. Luh and C.-Y. Lin. Optimal design of truss-structures using particle swarm optimization. *Computers & Structures*, 89(23):2221 – 2232, 2011.

D. Lukáš. Shape optimization of homogeneous electromagnets. In U. van Rienen, M. Günther, and D. Hecht, editors, *Scientific Computing in Electrical Engineering*, pages 145–152, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

S. W. Mahfoud. *Niching Methods for Genetic Algorithms*. PhD thesis, Champaign, IL, USA, 1995.

S. Martello and P. Toth. The 0-1 knapsack problem. In N. Christofides, A. Mingozzi, P. Toth, and C. Sandi, editors, *Combinatorial Optimization*, pages 237–279. John Wiley and Sons, 1979.

S. Martello and P. Toth. Upper bounds and algorithms for hard 0-1 knapsack problems. *Operations Research*, 45(5):768–778, 1997.

S. Martello and D. Vigo. Exact solution of the two-dimensional finite bin packing problem. *Management science*, 44(3):388–399, 1998.

G. B. Mathews. On the partition of numbers. *Proceedings of the London Mathematical Society*, s1-28(1):486–490, 1896.

G. Mavrotas, D. Diakoulaki, and A. Kourentzis. Selection among ranked projects under segmentation, policy and logical constraints. *European Journal of Operational Research*, 187(1):177–192, 2008.

M. Mavrovouniotis and S. Yang. Ant colony optimization with immigrants schemes for the dynamic travelling salesman problem with traffic factors. *Applied Soft Computing*, 13(10):4023–4037, 2013.

D. Merkle, M. Middendorf, and H. Schmeck. Ant colony optimization for resource-constrained project scheduling. *IEEE transactions on evolutionary computation*, 6(4): 333–346, 2002.

L. F. F. Miguel, R. H. Lopez, and L. F. F. Miguel. Multimodal size, shape, and topology optimisation of truss structures using the firefly algorithm. *Advances in Engineering Software*, 56:23 – 37, 2013.

S. Mirjalili and A. Lewis. S-shaped versus v-shaped transfer functions for binary particle swarm optimization. *Swarm and Evolutionary Computation*, 9:1–14, 2013.

S. Mirjalili and A. Lewis. Novel frameworks for creating robust multi-objective benchmark problems. *Information Sciences*, 300:158 – 192, 2015.

A. Misevicius. A tabu search algorithm for the quadratic assignment problem. *Computational Optimization and Applications*, 30(1):95–111, 2005.

A. Modiri and K. Kiasaleh. Modification of real-number and binary pso algorithms for accelerated convergence. *IEEE Transactions on Antennas and Propagation*, 59(1):214–224, 2011.

M. A. Mohammed, M. K. A. Ghani, R. I. Hamed, S. A. Mostafa, M. S. Ahmad, and D. A. Ibrahim. Solving vehicle routing problem by using improved genetic algorithm for optimal solution. *Journal of Computational Science*, 21:255–262, 2017.

K. Mombaur, A. Truong, and J.-P. Laumond. From human to humanoid locomotionan inverse optimal control approach. *Autonomous robots*, 28(3):369–383, 2010.

M. Monaci, U. Pferschy, and P. Serafini. Exact solution of the robust knapsack problem. *Computers & operations research*, 40(11):2625–2631, 2013.

T. Murata, H. Ishibuchi, and H. Tanaka. Multi-objective genetic algorithm and its applications to flowshop scheduling. *Computers & Industrial Engineering*, 30(4):957–968, 1996.

H. Naceur, Y. Guo, and J. Batoz. Blank optimization in sheet metal forming using an evolutionary algorithm. *Journal of Materials Processing Technology*, 151(1):183–191, 2004.

M. Naeem, U. Pareek, and D. C. Lee. Swarm intelligence for sensor selection problems. *IEEE Sensors Journal*, 12(8):2577–2585, 2012.

N. Noilublao and S. Bureerat. Simultaneous topology, shape and sizing optimisation of a three-dimensional slender truss tower using multiobjective evolutionary algorithms. *Computers & Structures*, 89(23):2531–2538, 2011.

N. Noilublao and S. Bureerat. Simultaneous topology, shape, and sizing optimisation of plane trusses with adaptive ground finite elements using moeas. *Mathematical Problems in Engineering*, 2013, 2013.

M. Ohsaki. *Optimization of finite dimensional structures*. CRC Press, 2016.

S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. *Journal of computational physics*, 79(1): 12–49, 1988.

I. Parmee and P. Hajela. *Optimization in industry*. Springer Science & Business Media, 2012.

D. Parrott and X. Li. Locating and tracking multiple dynamic optima by a particle swarm model using speciation. *IEEE Transactions on Evolutionary Computation*, 10 (4):440–458, Aug 2006.

C. Patvardhan, S. Bansal, and A. Srivastav. Solution of hard knapsack instances using quantum inspired evolutionary algorithm. *International Journal of Applied Evolutionary Computation (IJAEC)*, 5(1):52–68, 2014.

C. Patvardhan, S. Bansal, and A. Srivastav. Quantum-inspired evolutionary algorithm for difficult knapsack problems. *Memetic Computing*, 7(2):135–155, 2015.

M. A. A. Pedrasa, T. D. Spooner, and I. F. MacGill. Scheduling of demand side resources using binary particle swarm optimization. *IEEE Transactions on Power Systems*, 24 (3):1173–1181, 2009.

E. Pérez, F. Herrera, and C. Hernández. Finding multiple solutions in job shop scheduling by niching genetic algorithms. *Journal of Intelligent manufacturing*, 14(3-4):323–339, 2003.

R. Perez and K. Behdinan. Particle swarm approach for structural design optimization. *Computers & Structures*, 85(19):1579–1588, 2007.

A. Petrowski. A clearing procedure as a niching method for genetic algorithms. In *Proceedings of IEEE International Conference on Evolutionary Computation, 1996*, pages 798–803, May 1996.

D. Pisinger. *Algorithms for Knapsack Problems*. PhD thesis, Universitetsparken 1, DK-2100 Copenhagen, Denmark, 1995.

D. Pisinger. Where are the hard knapsack problems? *Computers and Operations Research*, 32(9):2271–2284, 2005.

B. Qu, J. Liang, and P. Suganthan. Niching particle swarm optimization with local search for multi-modal optimization. *Information Sciences*, 197:131 – 143, 2012.

H. Rahami, A. Kaveh, and Y. Gholipour. Sizing, geometry and topology optimization of trusses via force method and genetic algorithm. *Engineering Structures*, 30:2360–2369, 2008.

S. S. Rao and S. S. Rao. *Engineering optimization: theory and practice*. John Wiley & Sons, 2009.

U. T. Ringertz. On topology optimization of trusses. *Engineering Optimization*, 9(3): 209–218, 1985.

J.-h. Rong, J.-s. Jiang, D.-h. YAN, and B. Xu. Bridge structure topology optimization with multiple constraints. *Engineering Mechanics*, 4:032, 2002.

J. Rönkkönen, X. Li, V. Kyrki, and J. Lampinen. A generator for multimodal test functions with multiple global optima. In *Simulated Evolution and Learning*, pages 239–248. Springer, 2008.

J. Rönkkönen, X. Li, V. Kyrki, and J. Lampinen. A framework for generating tunable test functions for multimodal optimization. *Soft Computing*, 15(9):1689–1706, 2011.

T. A. Runkler. Ant colony optimization of clustering models. *International Journal of Intelligent Systems*, 20(12):1233–1251, 2005.

L. Sahoo, A. K. Bhunia, and P. K. Kapur. Genetic algorithm based multi-objective reliability optimization in interval environment. *Computers & Industrial Engineering*, 62(1):152–160, 2012.

H. Samarghandi and K. Eshghi. An efficient tabu algorithm for the single row facility layout problem. *European Journal of Operational Research*, 205(1):98–105, 2010.

R. Sedaghati, A. Suleman, and B. Tabarrok. Structural optimization with frequency constraints using the finite element force method. *AIAA Journal*, 40(2):382–392, 2002.

L. K. Seok and G. Z. Woo. A new structural optimization method based on the harmony search algorithm. *Computer Structure*, 82:781 – 798, 2004.

F. Shahzad, A. R. Baig, S. Masood, M. Kamran, and N. Naveed. Opposition-based particle swarm optimization with velocity clamping (OVCPSO). In *Advances in Computational Intelligence*, pages 339–348. Springer, 2009.

P. Shelokar, V. K. Jayaraman, and B. D. Kulkarni. An ant colony approach for clustering. *Analytica Chimica Acta*, 509(2):187–195, 2004.

Q. Shen, J.-H. Jiang, C.-X. Jiao, G. li Shen, and R.-Q. Yu. Modified particle swarm optimization algorithm for variable selection in MLR and PLS model: QSAR studies of antagonism of angiotensin II antagonists. *European Journal of Pharmaceutical Sciences*, 22(2?3):145–152, 2004.

W. Sheng, X. Liu, and M. Fairhurst. A niching memetic algorithm for simultaneous clustering and feature selection. *IEEE Transactions on Knowledge and Data Engineering*, 20(7):868–879, 2008.

Y. Shi and R. Eberhart. A modified particle swarm optimizer. In *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*, pages 69–73, 1998.

J.-K. Shin, K.-H. Lee, S.-I. Song, and G.-J. Park. Automotive door design with the ulsab concept. using structural optimization. *Structural and Multidisciplinary Optimization*, 23(4):320–327, 2002.

O. M. Shir, C. Siedschlag, T. Bäck, and M. J. Vrakking. Niching in evolution strategies and its application to laser pulse shaping. In *International Conference on Artificial Evolution (Evolution Artificielle)*, pages 85–96. Springer, 2005.

A. Sinha, P. Malo, A. Frantsev, and K. Deb. Finding optimal strategies in a multi-period multi-leader–follower stackelberg game using an evolutionary algorithm. *Computers & Operations Research*, 41:374–385, 2014.

A. Sinha, P. Malo, and K. Deb. A review on bilevel optimization: from classical to evolutionary approaches and applications. *IEEE Transactions on Evolutionary Computation*, 2017.

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

N. J. A. Sloane. The on-line encyclopedia of integer sequences (oeis), Feb. 2017. URL https://oeis.org/A052294.

K. Socha. *Ant colony optimisation for continuous and mixed-variable domains*. VDM Publishing Saarbrücken, 2009.

P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. *KanGAL report*, 2005005:2005, 2005.

A. Sutradhar, G. H. Paulino, M. J. Miller, and T. H. Nguyen. Topological optimization for designing patient-specific large craniofacial segmental bone replacements. *Proceedings of the National Academy of Sciences*, 107(30):13222–13227, 2010.

H. A. Taboada, J. F. Espiritu, and D. W. Coit. Moms-ga: A multi-objective multi-state genetic algorithm for system reliability optimization design problems. *IEEE Transactions on Reliability*, 57(1):182–191, 2008.

K. Tai and S. Akhtar. Structural topology optimization using a genetic algorithm with a morphological geometric representation scheme. *Structural and Multidisciplinary Optimization*, 30(2):113–127, 2005a.

K. Tai and S. Akhtar. Structural topology optimization using a genetic algorithm with a morphological geometric representation scheme. *Structural and Multidisciplinary Optimization*, 30(2):113–127, 2005b.

W. Tang, L. Tong, and Y. Gu. Improved genetic algorithm for design optimization of truss structures with sizing, shape and topology variables. *International Journal for Numerical Methods in Engineering*, 62(13):1737–1762, 2005.

I. X. Tassopoulos and G. N. Beligiannis. Solving effectively the school timetabling problem using particle swarm optimization. *Expert Systems with Applications*, 39(5):6029 – 6040, 2012a.

I. X. Tassopoulos and G. N. Beligiannis. A hybrid particle swarm optimization based algorithm for high school timetabling problems. *Applied Soft Computing*, 12(11):3472 – 3489, 2012b.

R. Thomsen. Multimodal optimization using crowding-based differential evolution. In *Congress on Evolutionary Computation, CEC2004*, volume 2, pages 1382–1389, June 2004.

T. Ting, M. V. C. Rao, and C. K. Loo. A novel approach for unit commitment problem via an effective hybrid particle swarm optimization. *IEEE Transactions on Power Systems*, 21(1):411–418, 2006.

H.-K. Tsai, J.-M. Yang, Y.-F. Tsai, and C.-Y. Kao. An evolutionary algorithm for large traveling salesman problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 34(4):1718–1729, Aug 2004. ISSN 1083-4419.

G. A. Vignaux and Z. Michalewicz. A genetic algorithm for the linear transportation problem. *IEEE transactions on systems, man, and cybernetics*, 21(2):445–452, 1991.

D. Wang, W. H. Zhang, and J. S. Jiang. Truss optimization on shape and sizing with frequency constraints. *AIAA Journal*, 42(3):622–630, 2004.

D. Wang, D. Tan, and L. Liu. Particle swarm optimization algorithm: an overview. *Soft Computing*, pages 1–22, 2017.

L. Wang, X. Wang, J. Fu, and L. Zhen. A novel probability binary particle swarm optimization algorithm and its application. *Journal of Software*, 3(9):28–35, 2008a.

L. Wang, X. Wang, J. Fu, and L. Zhen. A novel probability binary particle swarm optimization algorithm and its application. *Journal of software*, 3(9):28–35, 2008b.

L. Wang, R. Yang, Y. Xu, Q. Niu, P. M. Pardalos, and M. Fei. An improved adaptive binary harmony search algorithm. *Information Sciences*, 232:58–87, 2013.

M. Y. Wang, X. Wang, and D. Guo. A level set method for structural topology optimization. *Computer methods in applied mechanics and engineering*, 192(1):227–246, 2003.

Y. Wang, Z. Luo, N. Zhang, and Z. Kang. Topological shape optimization of microstructural metamaterials using a level set method. *Computational Materials Science*, 87:178 – 186, 2014.

D. P. Williamson and D. B. Shmoys. *The design of approximation algorithms*. Cambridge University Press, 2011.

C.-Y. Wu and K.-Y. Tseng. Truss structure optimization using adaptive multi-population differential evolution. *Structural and Multidisciplinary Optimization*, 42(4):575 – 590, 2010.

Y. Wu, Q. Li, Q. Hu, and A. Borgart. Size and topology optimization for trusses with discrete design variables by improved firefly algorithm. *Mathematical Problems in Engineering*, 2017, 2017.

G. Wscher, H. Haubner, and H. Schumann. An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3):1109 – 1130, 2007.

D. Xiao, X. Liu, W. Du, J. Wang, and T. He. Application of topology optimization to design an electric bicycle main frame. *Structural and Multidisciplinary Optimization*, 46(6):913–929, 2012.

Y. Xiao and A. Konak. A genetic algorithm with exact dynamic programming for the green vehicle routing & scheduling problem. *Journal of Cleaner Production*, 167:1450–1463, 2017.

Y. M. Xie and G. P. Steven. A simple evolutionary procedure for structural optimization. *Computers & structures*, 49(5):885–896, 1993.

Yang and Xin-She. *Nature-inspired metaheuristic algorithms*. Luniver press, 2010.

J. Yang, H. Zhang, Y. Ling, C. Pan, and W. Sun. Task allocation for wireless sensor network using modified binary particle swarm optimization. *IEEE Sensors Journal*, 14 (3):882–892, 2014.

Z. Ye, Z. Li, and M. Xie. Some improvements on adaptive genetic algorithms for reliability-related applications. *Reliability Engineering & System Safety*, 95(2):120–126, 2010.

Y. Zhang, S. Wang, and G. Ji. A comprehensive survey on particle swarm optimization algorithm and its applications. *Mathematical Problems in Engineering*, 2015, 2015.

M. Zhou and G. Rozvany. On the validity of eso type methods in topology optimization. *Structural and Multidisciplinary Optimization*, 21(1):80–83, 2001.

S. Zhou, W. Li, and Q. Li. Level-set based topology optimization for electromagnetic dipole antenna design. *Journal of Computational Physics*, 229(19):6915–6930, 2010.

J.-H. Zhu, W.-H. Zhang, and L. Xia. Topology optimization in aircraft and aerospace structures design. *Archives of Computational Methods in Engineering*, 23(4):595–622, 2016.

X. Zhu, Q. Yu, and X. Wang. A hybrid differential evolution algorithm for solving nonlinear bilevel programming with linear constraints. In *Cognitive Informatics, 2006. ICCI 2006. 5th IEEE International Conference on*, volume 1, pages 126–131. IEEE, 2006.

D. Zou, L. Gao, S. Li, and J. Wu. Solving 0-1 knapsack problem by a novel global harmony search algorithm. *Applied Soft Computing*, 11(2):1556–1564, 2011a.

D. Zou, L. Gao, S. Li, and J. Wu. Solving 0-1 knapsack problem by a novel global harmony search algorithm. *Applied Soft Computing*, 11(2):1556–1564, 2011b.

W. Zuo, T. Xu, H. Zhang, and T. Xu. Fast structural optimization with frequency constraints by genetic algorithm using adaptive eigenvalue reanalysis methods. *Structural and Multidisciplinary Optimization*, 43(6):799–810, 2011.

X. Zuo, C. Chen, W. Tan, and M. Zhou. Vehicle scheduling of an urban bus line via an improved multiobjective genetic algorithm. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):1030–1041, 2015.