Hybrid truss layout optimization for generating multiple design alternatives

by

Lazlo Bleker

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday January 22nd, 2021 at 14:30.

Thesis committee:

	Dr. ir. M.A.N. Hendriks	TU Delft, Chairman
	Dr. ir. M. Langelaar	TU Delft
	Ir. L.P.L. van der Linden	TU Delft

TU Delft

Preface

This thesis is the culmination of my master's degree in Civil Engineering at the Delft University of Technology. Reflecting on the last two years I realize I have learned so much not only, but especially, in the to me ever-fascinating world of topology optimization. Despite its lengthy academic history however, its applications in civil engineering practice are still few and far between. With the current development of an evermore increasing emphasis on sustainability it seems as though continuing to not utilize the power of topology optimization would be a missed opportunity. While there exist a multitude of reasons for the scarce use of topology optimization in civil engineering practice today, it would be an insurmountable task to address all of these at once. It is for this reason that I have decided to focus my efforts on one of the issues with current topology optimization methods, namely the lack of design alternatives, as well as to focus on one area of civil engineering practice, being 3-dimensional truss design.

I want to express my gratitude to the members of my thesis committee, Dr. ir. M.A.N. Hendriks, Dr. ir. M. Langelaar and Ir. L.P.L. van der Linden for their valuable advice and the interest they showed in my work. On more than one occasion have they offered me a key insight or suggestion that altered the course of my research for the better.

When I started with my research I had the intention of reducing the gap that exists today between research in topology optimization and practical civil engineering design. I hope I have contributed a small step towards that goal.

Lazlo Bleker
Delft, December 2020

Abstract

As of writing this thesis little research has been done in truss layout optimization for multiple design alternatives. A hybrid scheme is proposed in which the fast gradient-based search of the Ground Structure Method (GSM) is employed for the search of optimal topology and size, while a population-based meta-heuristic algorithm explores the non-convex parameter space of geometry optimization. The scheme works by employing a meta-heuristic algorithm to optimize the nodal coordinates of a truss structure where for each iteration and each member of the population a small-scale (i.e. problems with a small number of degrees of freedom) GSM is performed, in order to obtain the optimal topology for the given nodal locations. Three variants of the scheme based on three different meta-heuristic algorithms are developed: Firstly, an Artificial Bee Colony (ABC) and Particle Swarm Optimization (PSO) variant are developed in combination with an original topology identification method, to extract design alternatives from the solutions found throughout the optimization process. The developed topology identification method identifies different design alternatives based on topological differences by making use of existing graph isomorphism testing algorithms from graph theory. Additionally, the topology identification method employs a multi-step filtering process to prevent members which are non-critical to the performance of the structure to influence the design alternative selection process. Finally, a Multi-Species Particle Swarm optimization (MSPSO) variant is developed without the need for a topology identification method.

For all methods convergence speed in terms of number of iterations, topological variety and computation time per iteration have been evaluated and are compared. Of the developed methods, the ABC variant converges fastest towards a single good solution, however the topological variety is lacking. The slower converging MSPSO variant produces solutions with moderate topological variety, as well as reasonable material volumes. The PSO variant requires the least computation time per iteration of the developed methods. Its produced topological variety is closer to that of the ABC than the MSPSO variant and it has a slightly faster convergence speed than the MSPSO variant.

Direct usage of the MOSEK API is made instead of the more commonly used CVXPY API which reduces GSM problem setup times for small-scale (28 degrees of freedom) problems by a factor of 13. Computation times for the hybrid method, from start to finish, for 3D structures with 4 to 8 movable nodes (12 to 24 geometric degrees of freedom) range from 15 to 50 seconds on standard desktop PC hardware. Because of the multiple design alternative nature of the hybrid method, and consequently the end-user does not require fast back-to-back optimization runs, these computation times are deemed acceptable.

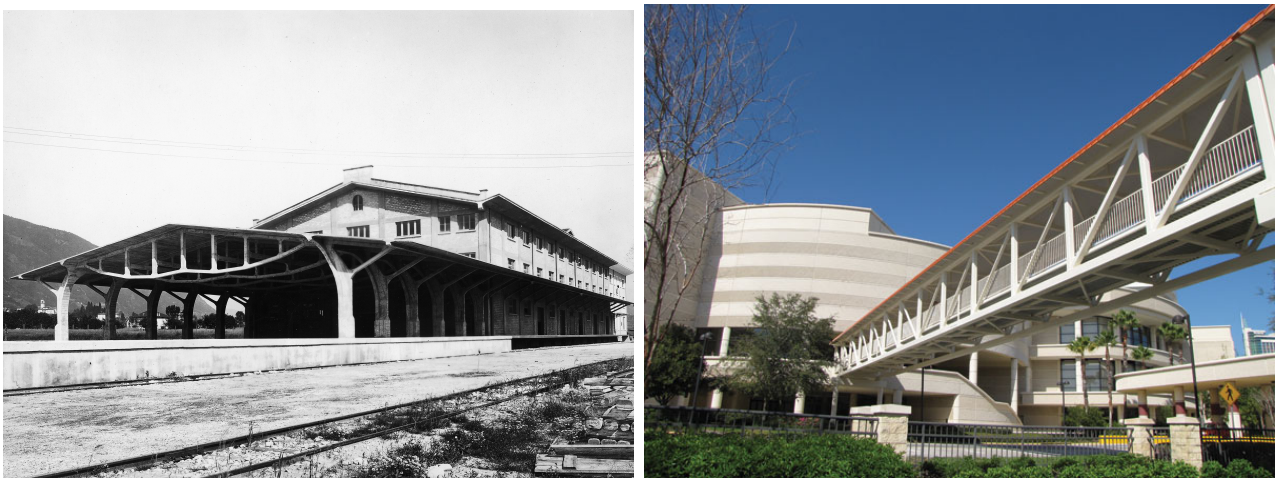
It is concluded that while in its current state the methods based on the hybrid scheme are unlikely to be suitable for usage in practice, further developments in methods to distinguish design alternatives could make the hybrid scheme, in particular the ABC variant, useful in the design of more material efficient truss structures.

Contents

Chapter 1

Introduction

The design of truss structures has undergone significant changes in the last century. At the start of the century structures were being designed with material efficiency being a primary concern. Due to the high cost of material, the material efficiency of structures was a large factor in the overall cost. As material has become cheaper, and labour has become more expensive, newly designed truss structures have reflected this change: Standardized structures such as Pratt, or Warren Trusses have become prevalent while custom truss designs have been reserved for very large scale projects or clients willing to pay a premium for architectural novelty. These standardized designs favour ease of construction over material efficiency, and are thus in the current climate more cost-effective than more intricate custom designs. There are two key reasons which suggest this development from favouring custom to favouring standardized structures could reverse in the near future: firstly, the cost of material is expected to increase. The world is becoming increasingly more aware of the influence of human activities on the environment. As carbon taxes and other environmental regulations become commonplace the price of the most common building materials, concrete and steel, are going to increase as well. Secondly, the construction of complex structures could become significantly cheaper, due to the emergence of large-scale additive manufacturing technologies.



(a) Magazzini Generali, Chiasso (1924) (b) Rosen centre foot bridge, Orlando (2011)

Figure 1.1: Classical (a) vs. Modern (b) truss design

During the transition from designing for material efficiency to designing for ease of construction, technology has not stood still. The introduction of technologies such as CAD and FEM have had a profound impact on the structures that can realistically be designed by engineers and architects. One technology however that has emerged during the era of designing for ease

of construction, which has not seen the wide adoption CAD and FEM have seen, is "structural topology optimization", or simply "topology optimization". Topology optimization is a technology which allows structures to be generated which are highly optimized for material usage. When material efficiency once again becomes a primary concern, Structural optimization, or "layout optimization" as it is often referred to when applied to truss structures could have a similar magnitude of impact on the design of (truss) structures as CAD and FEM once had.

1.1 A need for design alternatives

Despite the rich academic history of structural optimization, it has had relatively little impact on structural engineering in practice [Mueller, 2014]. There are many factors contributing to this, including shortcomings of current layout optimization methods. One major reason for this disconnect between research and practice, is that the goal of optimization has nearly always been to solely identify the global optimum. This approach could only work if the objective function perfectly encapsulated all the subjective qualities on which engineers and architects evaluate design alternatives (e.g. constructability or aesthetic qualities). Since truss design in structural engineering rarely occurs in absence of architectural goals this is not often the case. A more practical approach is to consider instead a selection of design alternatives. This way the end-user can opt for a design alternative based on the subjective qualities that are difficult to implement in the objective function of the optimization method.

Most commonly, layout optimization methods are based on gradient-descent in which the solution is found by moving in the direction of steepest descent in a convex search space. The advantage of such methods is that due to their convexity they are computationally inexpensive, while at the same time being very reliable. Because of the same convexity however, there is no obvious way to alter this method such that it is able to find multiple high quality solutions as design alternatives, since there exist no remarkable points in a convex search space other than the global optimum.

An alternative to a gradient-based method is a meta-heuristic method. These methods are based on trial-and-error search and are able to handle non-convex search spaces. The non-convexity of the search space is often considered a disadvantage since it allows the method to become trapped in a local optimum. For the purpose of finding multiple solutions however, the local minima in a non-convex search space provide good candidates for design alternatives. Compared to gradient-based methods, meta-heuristic methods are computationally expensive, making these methods unfit for problems with a considerable number of degrees of freedom.

A hybrid method that combines gradient descent and a trial-and-error approach could be a solution to finding multiple design alternatives while still keeping computation times under control.

1.2 Proposed method

There exist three basic types of structural optimization: size, geometry and topology optimization [Bendsøe and Sigmund, 2003]. In the context of truss layout optimization, size optimization changes the cross-sectional areas of the truss members while keeping the geometry and topology constant. Geometry optimization has as its parameters the spatial coordinates of the nodes of the truss, however still the topology remains unchanged. Finally, in topology optimization the topology is allowed to change i.e. the connectivity between the nodes of the truss changes. The creation of new nodes or the deletion of existing ones is also part of topology optimization.

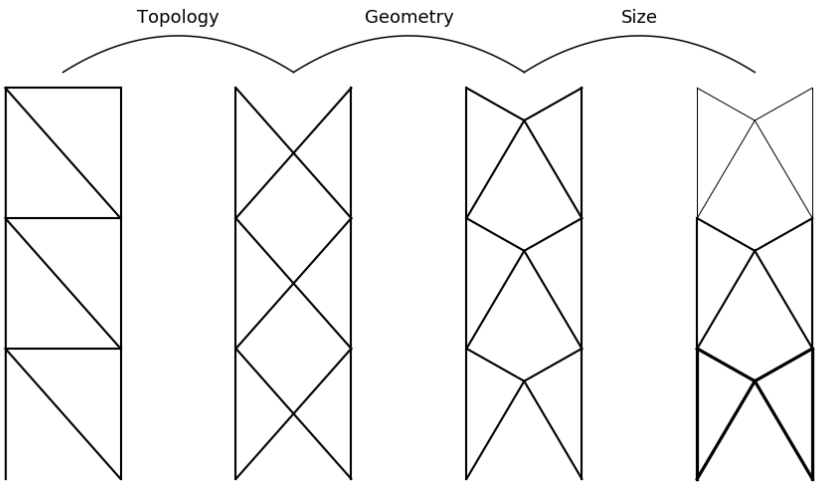


Figure 1.2: The three types of structural optimization for truss structures

To arrive at a global optimum for a particular problem it is insufficient to consecutively do a topology, geometry and size optimization. The optimal geometry for example does not only depend on the optimal topology, but this relation is true in reverse as well. To find a high quality solution it is therefore necessary that all three types of optimization are done concurrently.

A hybrid scheme is proposed in which the topology and size optimization are done through gradient-descent, based on the Ground Structure Method (GSM) approach. Concurrently the geometry optimization is handled by a meta-heuristic algorithm. A method based on this scheme would have a significantly lower computational cost than a purely meta-heuristic method due to most of the degrees of freedom being offloaded to the more efficient gradient-based approach. At the same time the problem formulation retains a non-convex search space allowing multiple local optima to be identified as design alternatives.

1.3 Research questions and scope

To explore the feasibility of the proposed hybrid scheme the main research question has been formulated:

"Is a method based on the proposed hybrid scheme capable of generating multiple good design alternatives?"

How good the design alternatives are is based on two factors: firstly the quality of the design alternatives is based on the extra material volume required as compared to a benchmark solution produced by a standard GSM approach. Secondly, a diverse range of design alternatives is desirable. Therefore design alternatives which are more distinct from one another are deemed to be better than design alternatives which are only slightly different.

The proposed hybrid scheme leaves room for different variants. Multiple methods based on the proposed scheme are be developed. In this thesis the following methods are considered:

1. An Artificial Bee Colony (ABC) method combined with a topology identification method.
2. A Particle Swarm Optimization (PSO) method combined with a topology identification method.

3. A Multi-Species Particle Swarm Optimization method (MSPSO).

Based on these methods the first sub-question to be answered is:

"Which of the considered meta-heuristic algorithms is best-suited for the proposed hybrid scheme?"

Furthermore it is the aim for the method to be applicable in engineering practice. It is therefore of importance that problems can be solved within reasonable time-frames.

"For what problem size can a method based on the proposed hybrid scheme produce results within a reasonable time-frame on a standard PC?"

Computation times will be measured on a PC equipped with an Intel Core i7-8700B processor (2018) and 16GB of RAM. This processor and memory size are comparable to what modern desktop PC's at engineering offices are equipped with and should therefore give a realistic estimate of computation times. Finally, the method will be developed for 3-dimensional problems as to increase its relevance for practical design applications.

1.4 Thesis structure

The structure of this thesis is as follows: in chapter 2 a literature review is presented on relevant developments in layout optimization, meta-heuristic optimization and hybrid optimization methods. Chapter 3 provides a detailed explanation of the proposed hybrid scheme and its different variants. In chapter 4 the different hybrid method variants are subjected to three case studies. Their results are compared in terms of convergence speed, design alternative variety and computation speed. Chapter 5 provides a discussion on the differences between the methods as well as the overall practicality of the proposed hybrid scheme. In chapter 6 the research questions are answered and suggestions for future research are given.

Chapter 2

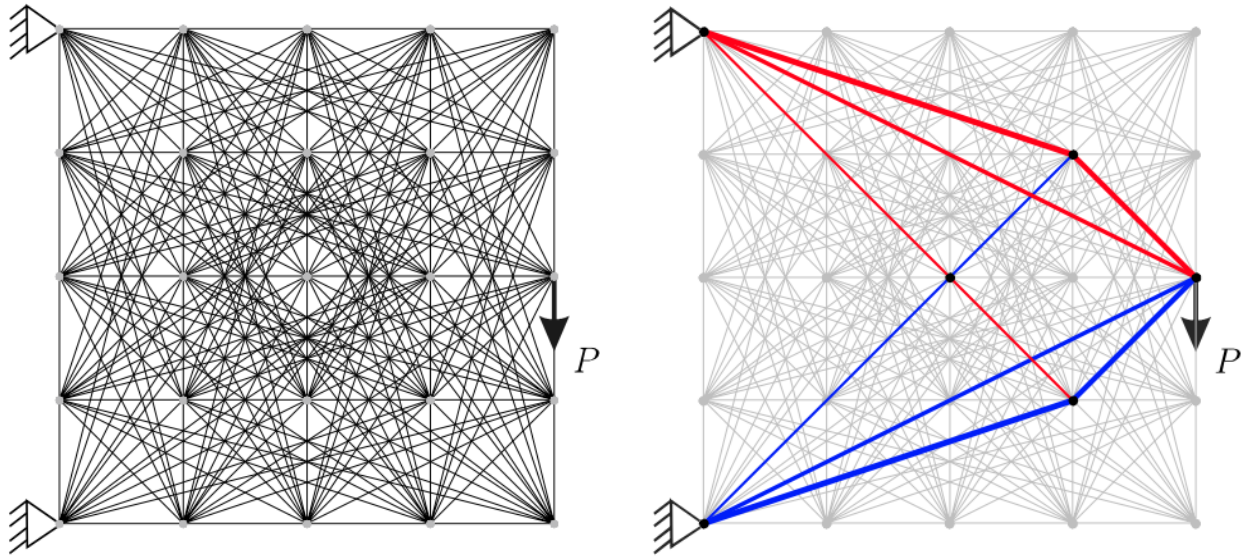
Literature review

This chapter presents existing work in the fields of layout optimization, meta-heuristic optimization and hybrid optimization, in general, but mostly as applied to truss structures.

The first paper on truss layout optimization was the 1904 paper by Michell [Michell, 1904]. In this paper the conditions for a truss structure to be optimal are given. Trusses that satisfy these conditions, and are thus optimal, are today known as Michell structures. However for nearly all problems it is impossible to solve the Michell problem analytically. For this reason computational methods for finding optimal truss layouts have been studied in recent decades. The most common objectives of these methods are to minimize compliance given a material volume constraint, or to minimize material volume given a compliance constraint. Both of these objectives result in identical topologies [Christensen and Klarbring, 2009]. It is possible to divide these computational methods into three categories: gradient-based, meta-heuristic and hybrid layout optimization methods.

2.1 Gradient-based layout optimization

Gradient-based optimization methods search for an optimal solution of the design space by moving a trial solution along the direction of steepest descent of the objective function. An example of a general-purpose gradient-based method is sequential quadratic programming (SQP). For truss optimization by far the most popular gradient-based method is the Ground Structure Method (GSM). Originally proposed by Dorn et al., the GSM is based on a ground structure consisting of a dense grid of nodes interconnected by truss members [Dorn et al., 1964]. The main design variables in the optimization procedure are the cross-sectional areas of the members. The objective function is to minimize the total material volume subject to some equilibrium and stress constraints. Although computationally expensive, it is desirable to have every node in the ground structure be connected to every other node (Figure 2.1). This is called a fully connected ground structure. Such a fully connected ground structure allows the largest number of possible solutions and thus, in general, a better optimal solution. Since its inception many additions to the original method have been proposed. To bypass the large computational cost of a fully-connected ground structure, an adaptive member adding scheme has been proposed, which makes the original method more efficient, particularly for large-scale problems [Gilbert and Tyas, 2003]. The modified method reduces the computational effort of large-scale problems by considering only a subset of all potential members as a starting point, after which additional members that could improve the solution are identified until no new potential members are left. Later on this method was extended to support multiple load cases as well [Pritchard et al., 2005].



(a) Ground structure (b) Optimal layout

Figure 2.1: Fully connected ground structure and accompanying optimal layout for a cantilever problem as found by the Ground Structure Method [L. He, 2019]

Fundamentally the GSM is a sizing optimization method, since its topology and geometry are static. However, if a fully connected ground structure is applied, and the cross-sectional areas of individual members are allowed to approach zero, the method effectively becomes a topology optimization method. Furthermore, if a dense enough grid of nodes is present in the ground structure, the GSM is a geometrical optimization method as well: the coordinates of any particular node can be altered by the vanishing of members connected to this node, and the reappearing of members connected to another node close to the position of the original node. This manifestation of geometry optimization does not come without complications however: Michell found that for many truss optimization problems the solution is a "truss-like continuum", which is a structural web of infinite members separated by infinitesimal spacing [Michell, 1904]. It is for this reason that structural optimization methods without any constraints on the complexity of the solution will often find solutions as complex as the ground structure allows. Since for an effective geometry optimization a dense ground structure is a prerequisite, this inevitably leads to solutions containing huge quantities of members (Figure 2.2). From a practical standpoint it is therefore of interest to somehow limit or penalize the complexity of the structure. The first attempt to introduce a constraint on the complexity of the solution was the addition of joint costs to the original GSM formulation [Parkes, 1975]. This method involves adding a material volume penalty to each member, related to its volume. Another variant of this concept is to have the material volume penalty be a constant [Prager, 1977]. An advantage of these simple complexity constraints is that their impact on the computational cost is negligible. A disadvantage however is that since effectively short members instead of the number of joints are penalized, its efficacy is found to be problem dependant [L. He, 2019], [Fairclough and Gilbert, 2020]. More recently it has been proposed to use the truss layout derived from the unconstrained optimization as a starting point for a geometrical optimization post-processing step involving the merging of nodes as well [He and Gilbert, 2015]. The results for large-scale structures are much closer to the Michell optimum structures when compared to joint cost methods, however its efficacy for small-scale structures has yet to be demonstrated. Another way to constrain the complexity of the structure is to enforce a limit on the number of joints and/or the minimum angle between connected members using a mixed integer linear programming formulation [Fairclough and Gilbert, 2020]. The method produces very high

quality solutions, however even for small-scale structures it boasts relatively high computation times. Continuously differentiable complexity constraints have also been developed although in contrast to the mixed integer formulation, these constraints are non-convex [Torii et al., 2016]. To deal with this non-convexity a gradient-based algorithm was used which starts from many different initial points in the design space.

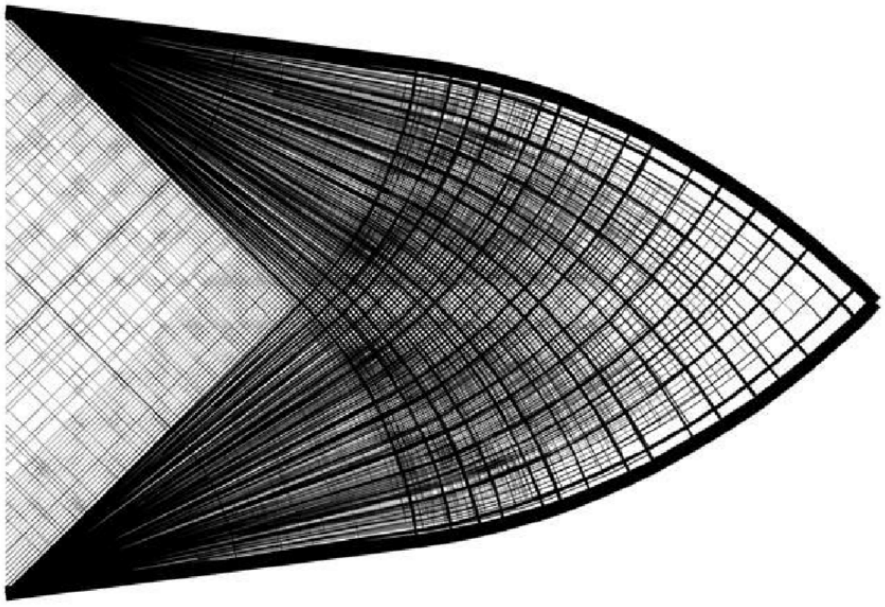


Figure 2.2: Optimal layout for a cantilever problem as found by the unconstrained Ground Structure Method [Gilbert and Tyas, 2003]

An alternative to the GSM has been proposed in the form of a "growth" method [Martínez et al., 2007]. This method is capable of generating high quality solutions in short computation times. The method allows a limit to be placed on the number of nodes which allows the solution to be constraint in complexity. A major disadvantage of the method is that it is only capable of treating a single load case and cannot easily be extended to treat multiple load cases [He and Gilbert, 2015].

2.2 Meta-heuristic layout optimization

As opposed to gradient-based optimization methods, meta-heuristic optimization methods look for an optimal solution based on trial and error. In general this leads to meta-heuristic optimization methods needing significantly more function evaluations, and thus more computation time, than gradient-based methods. On the other hand, meta-heuristic methods have the ability to escape local minima and thus do not necessarily require a convex problem formulation. Over the years a large number of methods have been proposed, many of them claiming superiority over other approaches. These algorithms can be classified into trajectory-based and population-based, in which the former is based on the random search of a single agent, and the latter employs multiple agents (a population) simultaneously. For both types it is the goal to maximize the so-called "fitness" of the agents. Popular meta-heuristic methods include Genetic Algorithms (GA) [Holland, 1975], Particle Swarm Optimization (PSO) [J. Kennedy, 1995], Differential Evolution (DE) [Storn and Price, 1997] and Artificial Bee Colony (ABC) [Karaboga and Basturk, 2007]. Additionally for most methods multiple variants exist. A comparison of a handful of methods for a large number of benchmark functions shows that no single algorithm performs best in the majority of benchmark functions [Ab Wahab et al., 2015]. It can also be

proven that the average performance of any pair of algorithms across all possible problems is identical [Wolpert and Macready, 1997]. This means that if one algorithm has superior performance over the other for a particular class of problems, the reverse must be true for some other class of problems.

Like there is a large range of different meta-heuristic optimization methods, there exist also many ways of handling inequality constraints by these methods. It is possible to categorize different constraint handling techniques as follows [Michalewicz, 1996]:

1. methods based on preserving feasibility of solutions;
2. methods based on penalty functions;
3. methods that make a clear distinction between feasible and infeasible solutions and
4. other methods.

Methods based on preserving feasibility of solutions rely on the optimization method generating only trial solutions which are within the feasible region. An advantage of this class of methods is that no time is spent by the algorithm on checking the feasibility of trial solutions. An example of such a method is GENOCOP, in which trial solutions are linear combinations of existing solutions [Michalewicz and Janikow, 1996]. The method requires a convex feasible region as well as an initial population entirely situated within the feasible region. Methods based on penalty functions consider the unconstrained optimization problem with some penalty added to the objective function for the infeasible part of the parameter space. Advantages of these methods are that in case of a non-convex, or even disjoint feasible region, the algorithm can take a solution path through the infeasible region from one part of the feasible region to another. A disadvantage is that some time is wasted searching for an optimum within the infeasible region. The severity of the applied penalty can be static [Homaifar et al., 1994] or change throughout the duration of the optimization [Joines and Houck, 1994]. Finally an example of a method that makes a clear distinction between feasible and infeasible solutions is Deb's method [Deb, 2000]. It functions by the use of a tournament selection operator where a feasible solution is always preferred over an infeasible one. When two infeasible solution are compared the one having a smaller constraint violation is preferred. Another technique in this category is to make infeasible solutions feasible by moving them to a near point on the feasibility boundary [Michalewicz and Nazhiyath, 1995]. An advantage of these methods is that, given that the initial population is entirely feasible, no time is wasted searching in the infeasible region. Disadvantages are that the methods perform worse on problems with a non-convex or disjoint feasible region as well as that feasibility has to be checked for every trial solution.

Many attempts have been made to apply meta-heuristic optimization to the search for optimal truss layouts. A modified version of the PSO algorithm has been applied to truss structures [Luh and Lin, 2011]. The ABC algorithm in conjunction with an adaptive penalty constraint handling method has been applied to truss structures as well [Sonmez, 2011]. A similar study in which a DE algorithm has been used instead has also been performed [Wu and Tseng, 2010]. A study in which GA was used to optimize truss structures subject to some complexity constraints has also been done [Wang and Ohmori, 2010]. A combination of different meta-heuristic methods has also been applied to truss structures [Maheri et al., 2016]. Due to the inherent computational demand of meta-heuristic optimization methods, all of the above named studies have been limited to small-scale truss structures with a very limited number of degrees of freedom. Most often the meta-heuristic algorithm is supplied with the parameters of the optimization directly, be it the cross-section size of members, the nodal coordinates, or both.

An alternative approach to this is the use of a "shape grammars". Originally introduced by Stiny and Gips [Stiny and Gips, 1972], shape grammars consist of a set of rules that alter an existing geometric shape. With this approach the parameters of the meta-heuristic optimization are not simply modifications of cross-section sizes or nodal coordinates but rather the application of one of the rules from the shape- or structural grammar. Due to the geometric nature of truss structures, shape grammars are suitable for and have been applied to meta-heuristic layout optimization techniques [Shea, 1997], [Hooshmand and Campbell, 2016]. Structural grammars have also been used to generate multiple design alternatives [Mueller, 2014]. While structural grammars lower the computational load as compared to traditional parameters for the meta-heuristic algorithm, they also limit the possible solution space. To obtain good results it is therefore often necessary to create new rules for every new problem.

2.3 Hybrid layout optimization

Hybrid optimization methods combine gradient-based and meta-heuristic methods with the aim to exploit advantages of both. Although when it comes to truss layout optimization these hybrid methods have been studied much less than pure gradient-based or meta-heuristic optimizations, some attempts have been made. In particular combining the local search speed of gradient-based methods and the global search ability of meta-heuristic methods is often the main goal of these hybrid methods. To exploit these respective strengths, most proposed methods are based on a two-phase framework in which a global search is performed by the meta-heuristic method in the first phase and a local search by the gradient-based method in the second phase. This framework has been applied for a combination of SQP and PSO [Plevris and Papadrakakis, 2011]. Another way to combine gradient-based and meta-heuristic methods is to embed gradient-based operators within a meta-heuristic global search. An example of such an approach is a method in which an optimality criteria scheme is embedded within a GA process [Zuo et al., 2014]. For both studies speedups compared to purely meta-heuristic approaches have been observed.

2.4 Research gap

Despite the large body of work in the field of layout optimization, its applications in engineering practice remain nearly non-existent. For some of the requirements for application in practice, numerous attempts to address them have been made. Support for multiple load cases as well as the implementation of complexity constraints are examples of practical requirements in which significant progress has been made. The desire for multiple design alternatives stemming from the more abstract requirements of real world design however, remains largely unaddressed (with the notable exception of Mueller's work on structural grammars). The aim of this thesis is to contribute to addressing this need for multiple design alternatives while retaining, or implementing in new ways, support for previously addressed requirements such as the possibility of multiple load cases and complexity constraints.

Chapter 3

Methods

This chapter provides a detailed description of the proposed hybrid scheme. Different elements of the scheme, composing of both existing methods as well as original contributions, are explained in the different sections.

3.1 Proposed hybrid scheme

The proposed hybrid scheme works as a meta-heuristic algorithm in which each function evaluation is replaced with a modified version of the GSM. The modified GSM nested within the meta-heuristic algorithm works the same way as a traditional GSM does. The difference is that instead of a ground structure with a dense grid of nodes, only a handful of nodes initialized at random locations within the design space are present. The number of nodes in the GSM is a parameter of the hybrid method and is an upper bound on the number of nodes present in the solution. The modified GSM functions as both the size and topology optimization of the hybrid method. It does not fulfill the role of geometry optimization, since the nodes are positioned sparsely within the design space and the GSM does not support the movement of nodes. Because the nodal coordinates are the parameters of the meta-heuristic algorithm, it serves the role of the remaining (non-convex) geometry optimization part of the hybrid scheme. Allocating the role of geometry optimization to a meta-heuristic algorithm comes with a computational cost, however it enables two key advantages: firstly, because the dense nodal grid has been replaced by a select number of nodes, this number of nodes automatically functions as a complexity constraint by limiting the possible number of nodes in the solution. Secondly, the search of the non-convex parameter space of the geometry of the structure allows for the discovery of multiple local minima which, when identified, can serve as design alternatives.

A simple 2-dimensional problem is presented to make the proposed method more clear. Figure 3.1 shows a simply supported span with two movable nodes. The nodes are only free to move in one dimension so that the problem has 2 degrees of freedom, which in turn allows the parameter space to be visualized as a surface. In general, movable nodes would be free to move in all available dimensions. For this case the parameters on the horizontal axes of the surface plot, x_1 and x_2, represent the horizontal coordinates of the movable nodes p_1 and p_2 respectively. Every point on the surface represents a unique combination of locations of the movable nodes. The height of the surface represents the material volume of the truss that is generated by the modified GSM for the nodal locations represented by that point on the surface.

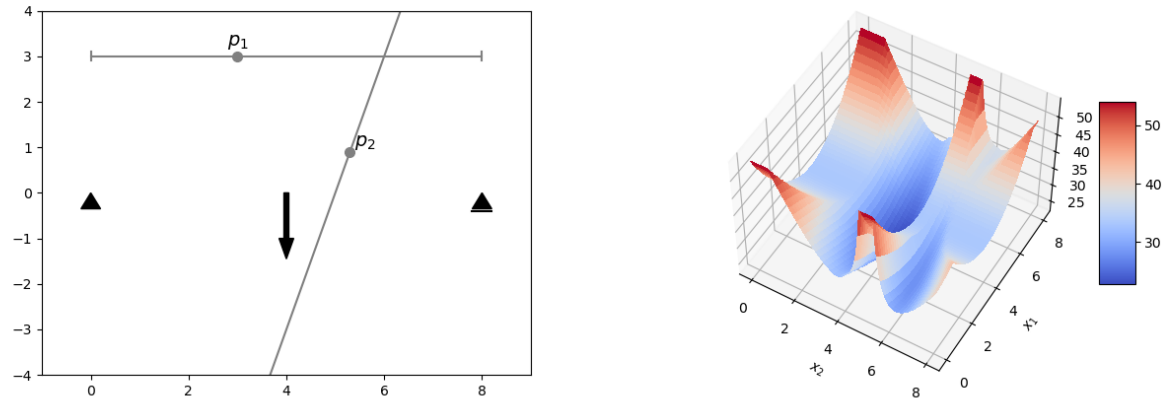


Figure 3.1: Simply supported span with 2 geometrical degrees of freedom (left) and the corresponding material volumes in the full parameter space (right)

Due to the nature of geometrical optimization problems the parameter space is non-convex, with more than one local minimum. Figure 3.2 shows a selection of truss structures which are generated for interesting points in the parameter space, such as local minima. The selected structures display the possibility of topological variety in design alternatives, even for such a simple problem setup.

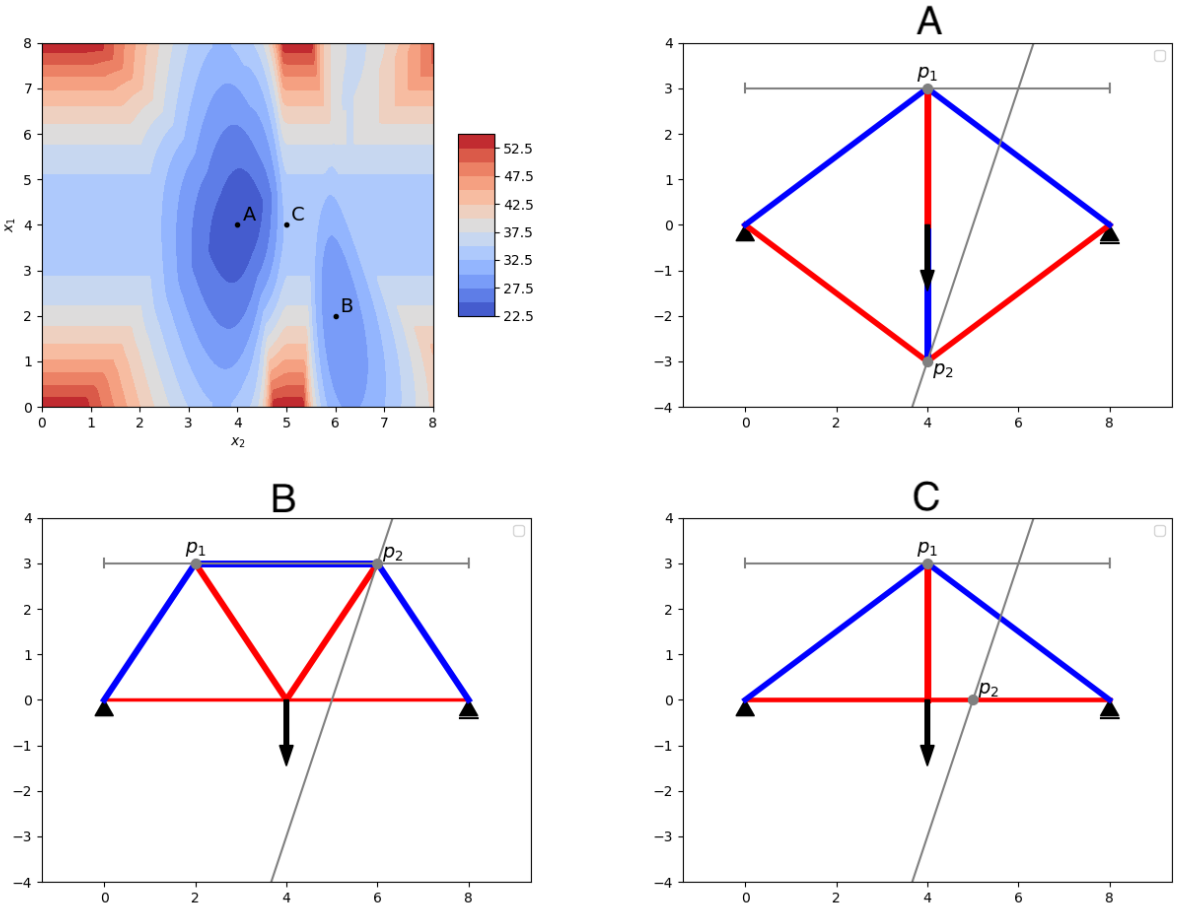


Figure 3.2: Simply supported span parameter space and the corresponding generated truss structures for three particular points in the parameter space

The parameter space is explored by a population-based meta-heuristic algorithm. Figure 3.3 gives an overview of one sample iteration. In this case each member of a population of size 10 updates its location so that it changes from the base of an arrow to its tip. Notice that not all arrows point in a direction in which the material volume becomes lower. The meta-heuristic algorithm allows individual members of the population to explore less optimal areas of the parameter space to allow for the possibility of escaping local minima. Figure 3.4 displays for one member of the population (shown as a red arrow in Figure 3.3) how the location of nodes of the truss is influenced by the change of location in the parameter space. It also demonstrates the impact of the geometrical change on the resulting structure generated by the modified GSM. In this case the perturbation in the parameter space resulted in a change in topology for the resulting structure.

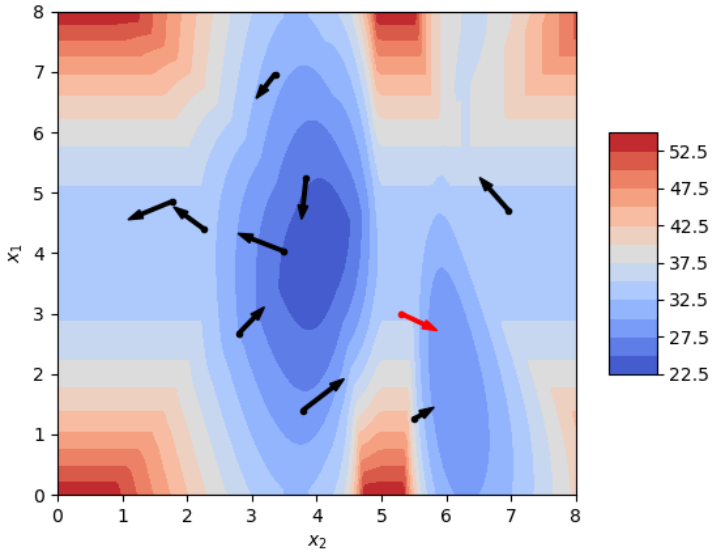


Figure 3.3: One iteration of a population-based meta-heuristic algorithm exploring the parameter space

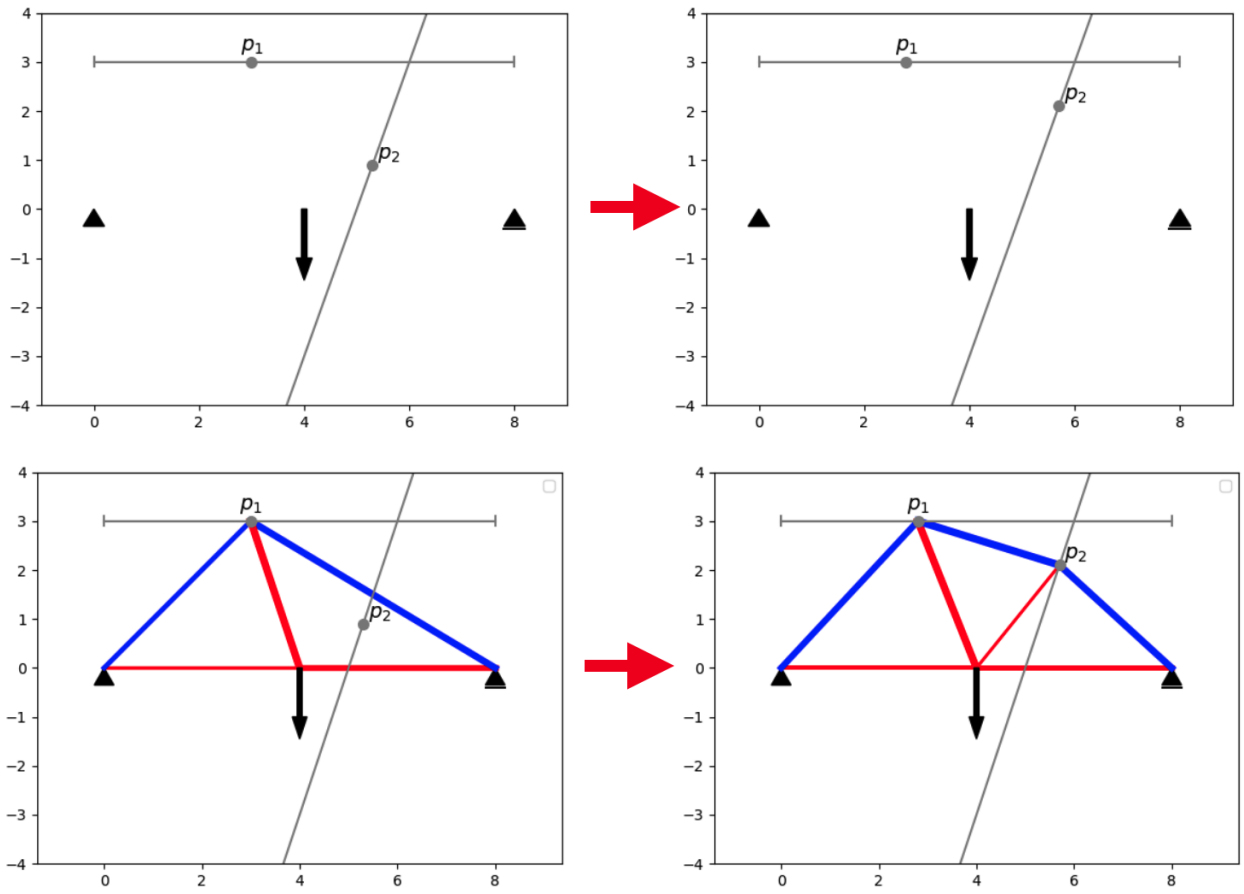


Figure 3.4: Effects on the topology of the structure of one population member from one geometry optimization iteration

The population keeps exploring the parameter space until a termination criterion, such as exceeding a limit on the number of iterations, is met. For two out of the three developed variants of the hybrid scheme, this optimization process is followed by a design alternative selection process in the form of a topology identification method. Figure 3.5 shows a diagram explaining the order in which different elements of each method are executed. In the following sections each element from the diagram will be explained in detail.

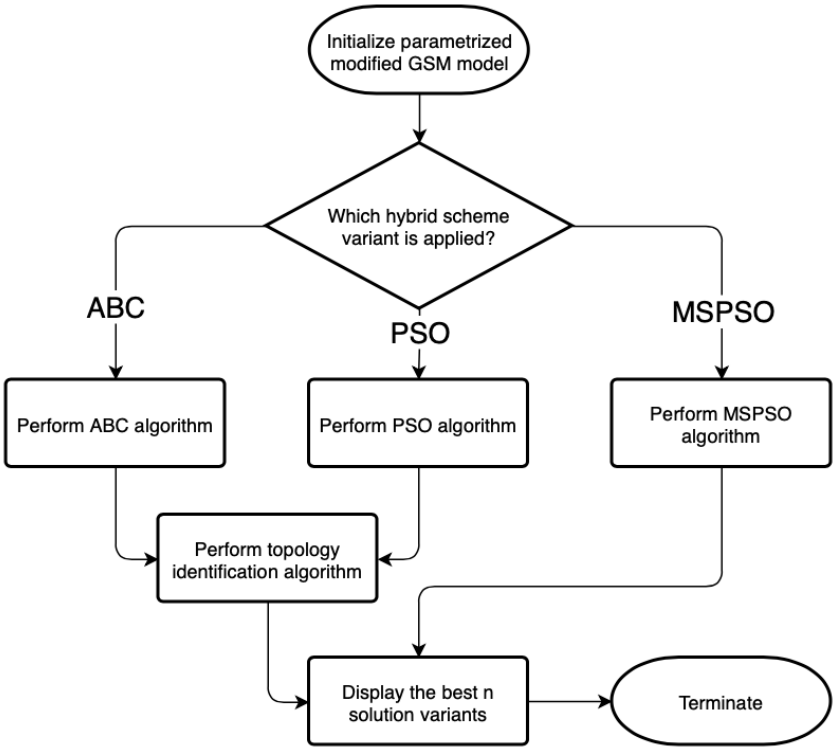


Figure 3.5: Diagram illustrating the high-level hybrid scheme workflow

3.2 Modified Ground Structure Method

The only change to the modified ground structure method as compared to the original method, is with respect to the arrangement of the nodes from the ground structure. Therefore its internal workings remain unchanged. Any existing GSM code would thus be compatible with the proposed hybrid scheme. An example of a publicly available piece of code written in Python by He et al. [L. He, 2019] boasts good performance for large-scale problems, as it was designed to do so. For small-scale problems however, the implementation becomes hugely inefficient. The cause for this is two-fold: firstly the implementation employs an adaptive member adding scheme. While this results in major speed-ups for large-scale problems, small-scale problems are actually slowed down by this approach. This problem can very simply be overcome by eliminating the adaptive member adding scheme. The second reason for the inefficiency of the script is that it makes use of a python package called CVXPY [Diamond and Boyd, 2016]. This package allows convex optimization problems to be written in a user-friendly manner while making use of highly efficient solver packages, such as in this case, MOSEK (`www.mosek.com`). The conversion of the input given to the CVXPY API to a form in which it is compatible with the MOSEK API does introduce some computational overhead. For large-scale GSM problems, and thus many degrees of freedom, this overhead is negligible compared to the time spent by MOSEK on solving the problem. For small-scale problems however, this overhead does become significant up to the point where the time spent on this conversion overhead can be equal to many times the solver time. This additional computational overhead can be overcome by directly interacting with the MOSEK API instead of indirectly via the CVXPY API. Doing this does however introduce additional requirements to how the optimization problem is formulated.

3.2.1 Primal formulation

Arguably the most intuitive formulation of the ground structure topology optimization problem is the primal form, in which the material volume is minimized while being subject to equilibrium equations and stress constraints. This primal formulation of the ground structure optimization problem, subject to multiple load cases, can be defined as follows [Bendsøe et al., 1994]:

$$\begin{aligned} \min_{A,S_{(l)}\in\boldsymbol{R}^M} \quad & V = \boldsymbol{L}^T\boldsymbol{A} \\ \text{s.t.} \quad & \boldsymbol{B}^T\boldsymbol{S}_l = \boldsymbol{P}_l \\ & -\boldsymbol{A}\sigma_C \le \boldsymbol{S}_l \le \boldsymbol{A}\sigma_T \\ & \text{for} \quad l = 1, 2, ..., K \\ & A_i \ge \max_{l=1,2,...,K}\left(\frac{S_{l,i}}{\sigma_T}, \frac{-S_{l,i}}{\sigma_C}\right) \quad \text{for} \quad i = 1, 2, ..., M \end{aligned} \tag{3.1}$$

where:

V = total material volume
$\boldsymbol{L}$ = vector of member lengths
$\boldsymbol{A}$ = vector of member cross-section areas
M = number of potential bars
$\boldsymbol{B}$ = equilibrium matrix
K = number of load cases
$\boldsymbol{S}_l$ = vector of member forces corresponding to load case l
$\boldsymbol{P}_l$ = external load vector corresponding to load case l
σ_C = maximum allowable compressive stress
σ_T = maximum allowable tensile stress

As of writing this thesis, MOSEK does not support a "maximum" operator outside of the maximization of the objective function. It is for this reason that only the single load case variant ($K = 1$) of the primal formulation can be implemented directly with MOSEK.

3.2.2 Dual formulation

Optimization problems can always be formulated from two different perspectives: the primal and the dual formulation [Boyd and Vandenberghe, 2004]. In general, the solution to the dual problem is a lower bound of the solution of the primal problem, and their difference is equal to the "duality gap". If however the problem is convex, as well as its constraint functions affine, then these are sufficient conditions for the duality gap to be equal to zero. As these conditions apply to the ground structure optimization problem, the duality gap is equal to zero and thus the solution to the dual problem equal to the solution of the primal problem. The dual formulation is given by [Sokól, 2014]:

$$\begin{aligned} \max_{\boldsymbol{u}_l\in\boldsymbol{R}^N,\boldsymbol{e}_l^+,\boldsymbol{e}_l^-\in\boldsymbol{R}^M} \quad & W = \sum_l \boldsymbol{P}_l^T\boldsymbol{u}_l \\ \text{s.t.} \quad & \sum_l(\sigma_T\boldsymbol{e}_l^+ + \sigma_C\boldsymbol{e}_l^-) = \boldsymbol{L} \\ & \boldsymbol{B}\boldsymbol{u}_l - \boldsymbol{e}_l^+ + \boldsymbol{e}_l^- = 0 \\ & \boldsymbol{e}_l^+ \ge 0, \quad \boldsymbol{e}_l^- \ge 0 \\ & \text{for} \quad l = 1, 2, ..., K \end{aligned} \tag{3.2}$$

where:

W = sum of external work (compliance)
$\boldsymbol{u}_l$ = vector of nodal displacements corresponding to load case l
$\boldsymbol{e}_l^+$ = vector of member elongation
$\boldsymbol{e}_l^-$ = vector of member compression

This formulation, albeit less intuitive, lacks the maximization operator in the constraint,s and can thus be directly implemented using the MOSEK API.

3.2.3 Parametrization

Conventional GSM scripts are designed to solve any given problem only once. Since the method is fully deterministic, there is no reason to solve a problem multiple times. In the context of the proposed hybrid scheme however, many slight variations of the same problem are solved after one another. Due to the similarity between the consecutive GSM runs, it is possible to save computation time by setting up a parametrized problem only once, after which it is used for a number of times equal to the population size of the meta-heuristic algorithm times the number of iterations. In this parametrized model only those variables which are subject to change are parameters, while all other variables are set up as constants. The member length vector $\boldsymbol{L}$ and equilibrium matrix $\boldsymbol{B}$ for example, are dependent on the nodal locations. Since the nodal locations are dynamic in between consecutive runs of the GSM subroutine, $\boldsymbol{L}$ and $\boldsymbol{B}$ are both set up as parameters. Their size, as well as the sparsity pattern of $\boldsymbol{B}$, are however static, and are thus set up as constants. Finally, the variables independent of nodal locations, $\boldsymbol{P}_l$, σ_C and σ_T can all be set up as constants.

3.3 Meta-heuristic algorithm

The modified GSM from the previous section serves as the fitness function of the meta-heuristic algorithm. The input for the modified GSM, and thus the parameters of the meta-heuristic algorithm are the spatial coordinates of the nodes of the truss. Thus the dimensionality of the parameter space of the meta-heuristic algorithm is defined by

$$D = d * n - f \tag{3.3}$$

where:

D = number of dimensions of the parameters space
d = number of dimensions of the design space
n = number of nodes
f = number of fixed degrees of freedom

The output of the modified GSM is the total material volume V. In order to preserve the convention of maximizing fitness, while still minimizing V, the following relation is employed:

$$f = \frac{1}{1+V} \tag{3.4}$$

in which f is the fitness value and V the material volume. This function ensures that a lower material volume corresponds to a larger fitness value, which represents a better solution.

For all variants, boundaries of the design space are dealt with by converting infeasible solutions to feasible ones. This is done by moving violating degrees of freedom to the value of the closest boundary. This method is suitable for this type of problem first of all because the parameter space is entirely convex, so there would be very little benefit in making use of a penalty function. Secondly, the fitness function (modified GSM) is fairly computationally expensive. This makes it so that the extra computation time required for checking feasibility of solutions becomes insignificant in the total computation time of the method.

3.3.1 Artificial Bee Colony

The Artifical Bee Colony (ABC) algorithm is a population-based meta-heuristic optimization method. First conceived by Karaboga, the algorithm is inspired by the division of labor exhibited by honey bees [Karaboga, 2005]. The population is split in two, with one half becoming "employed bees", while the other half is assigned the task of "onlooker bees". These two groups represent two distinct phases in each iteration of the algorithm (Figure 3.6).

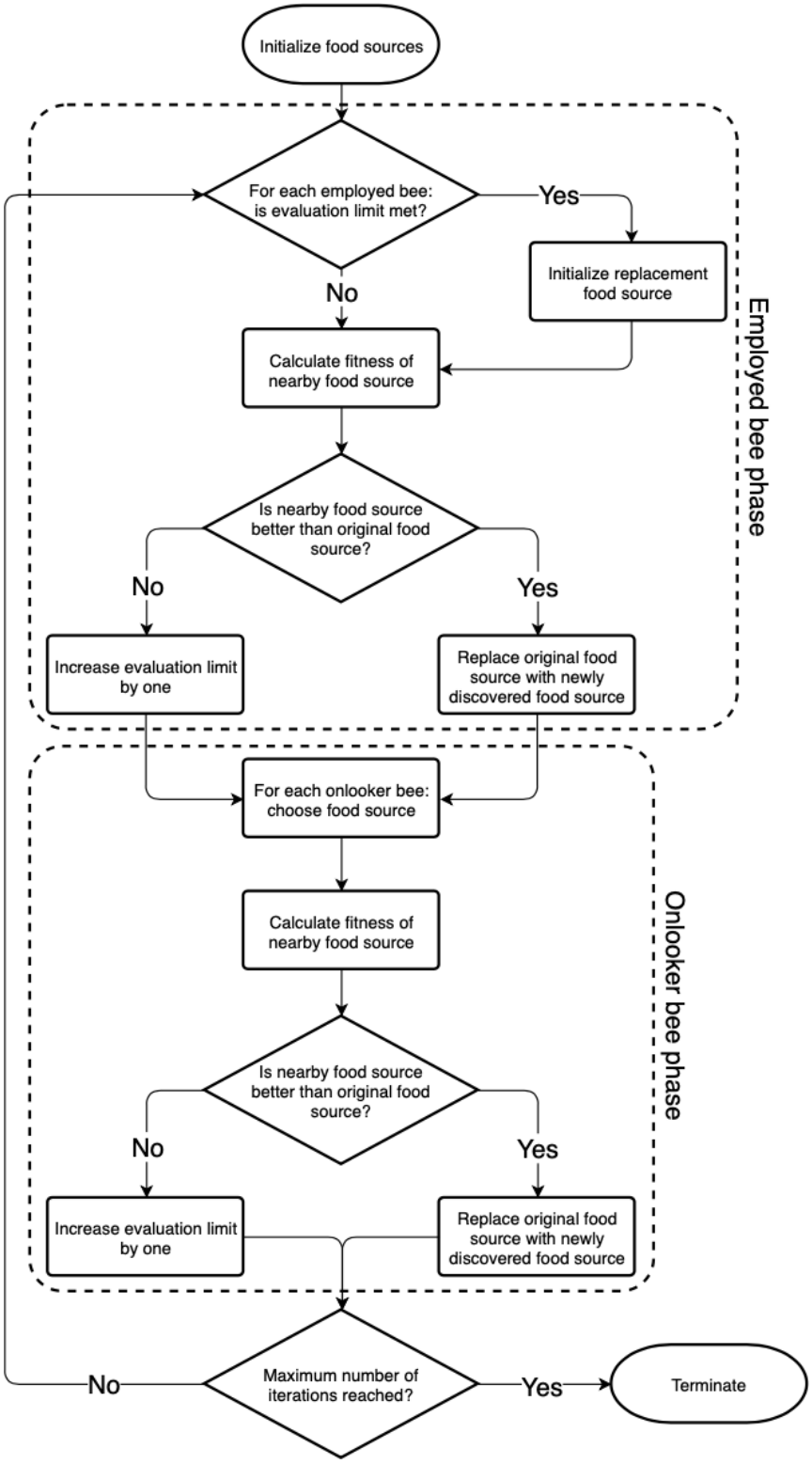


Figure 3.6: Diagram illustrating the artificial bee colony workflow

The algorithm starts off with initializing and assigning a "food source" to each employed bee. In this context, a food source simply represents a location in the parameter space. After initialization the employed bees phase starts with each employed bee first checking if it has reached the evaluation limit (more on that later). If the evaluation limit has not been reached, the fitness of a nearby food source is evaluated. The nearby food source is determined by

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \tag{3.5}$$

where:

v = nearby food source
x = original food source
ϕ = random number between -1 and 1
i = index of current food source
j = random index of parameter space dimension
k = random index of any food source that is not i

The perturbation from x to v is dependent on the distance between x and another randomly chosen food source. This ensures that step sizes remain a reasonable size for the search space that is being explored. The newly found food source is evaluated and if it has a better fitness value than the original, it replaces it, with the employed bee now being assigned to the new food source. If the newly found food source has a worse fitness value, it is discarded and the evaluation counter of the original food source is increased by one.

Once all employed bees have attempted to improve their respective food source, the onlooker bees phase starts. Every onlooker bee chooses a food source depending on its associated probability value. This value is calculated by

$$p_i = \frac{f(x_i)}{\sum_{n=1}^{SN} f(x_n)} \tag{3.6}$$

where:

p = probability value
f = fitness function
SN = number of food sources

Since the probability value is higher for higher fitness values, the onlooker bees are distributed such that the best food sources are most frequently chosen. Once an onlooker bee has chosen a food source its behaviour is identical to that of the employed bee: a nearby food source is generated according to (3.5). If this new food source is better than the original, the employed bee of the original food source is moved to the newly discovered food source. If the new food source is worse than the original, the evaluation counter is increased once more. Once all onlooker bees have generated a new food source, the onlooker bees phase ends and given that the maximum number of iterations has not yet been reached, a new iteration continues this cycle of alternating phases.

After the first iteration, it becomes possible that at the start of the employed bees phase one of the food sources has met the evaluation limit. Once this happens, before a nearby food source is generated, the original food source is replaced with a newly initialized food source in a random location in the parameter space, and its evaluation counter is set to zero. This mechanism allows food sources which are likely unimprovable (due to being stuck in a local minimum for example) to be abandoned. The evaluation limit of unsuccessful attempts to improve the food source before it is abandoned is a parameter of the method. A low evaluation limit allows the algorithm to encounter many different local minima, at the cost of some solution quality. For finding many design alternatives this could be a valuable feature, which is why it was chosen as one of three alternatives to be implemented.

3.3.2 Particle Swarm Optimization

The Particle Swarm Optimization (PSO) algorithm is one of the earliest, and still one of the most popular population-based meta-heuristic algorithms. Introduced by Kennedy and Eberhart, it was originally intended to simulate social behaviour in groups of animals [J. Kennedy, 1995]. Despite its age, perhaps due to its simplicity, PSO still performs very well when compared with other more recently introduced algorithms [Ab Wahab et al., 2015]. PSO works by a population of particles exploring the parameter space, in which their movement is guided by the personal experience of each particle, but also by the swarm as a whole. Figure 3.7 shows a diagram displaying the workflow of the algorithm.

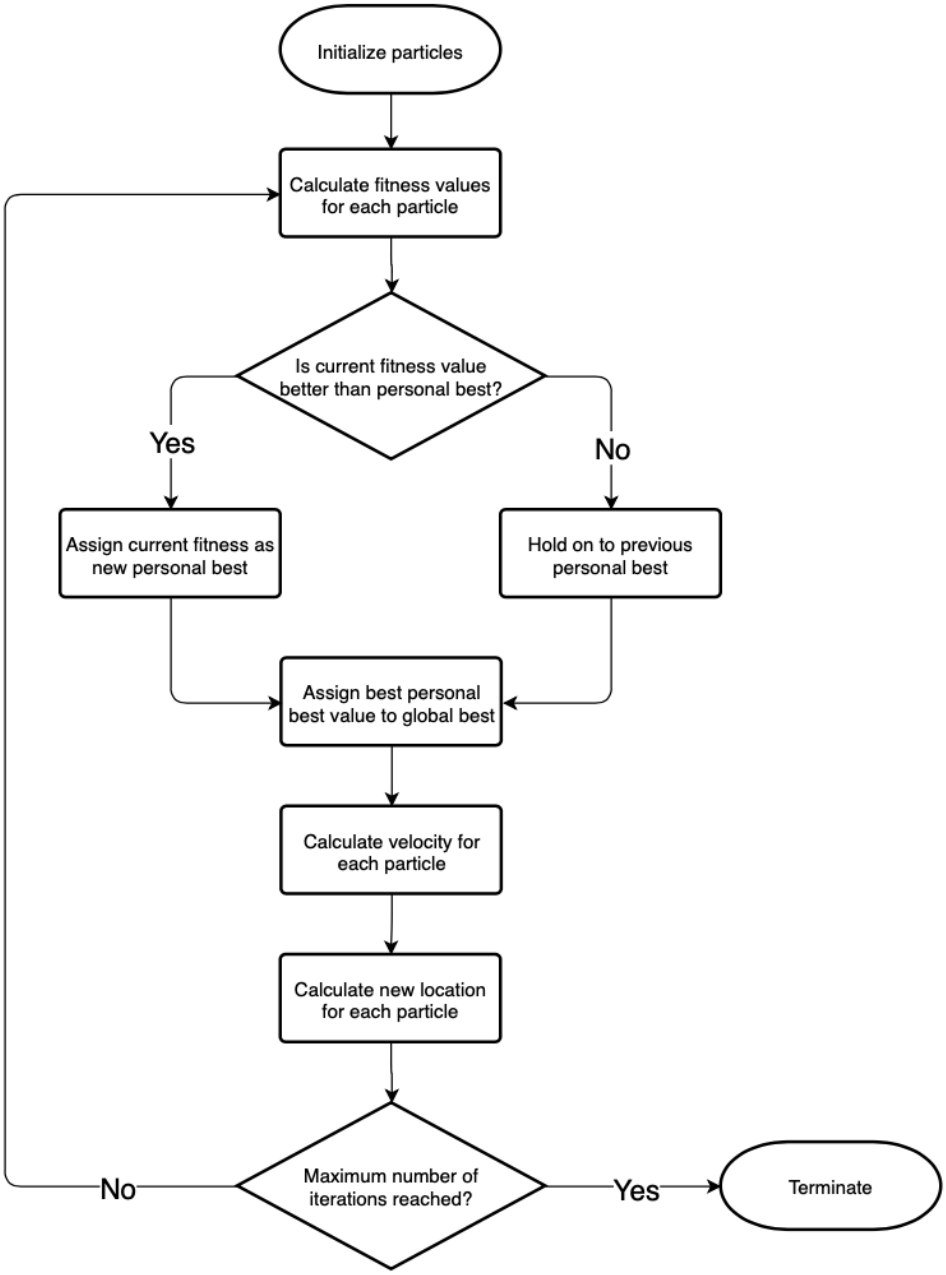


Figure 3.7: Diagram illustrating the particle swarm optimization workflow

The algorithm starts with initializing the location, as well as a velocity, for each particle in the swarm. For each particle their fitness value is evaluated as well. Every particle keeps track of their personal best position, in which the best position is the position corresponding to

the highest fitness value. A global variable, called the global best, represents the best personal best position of all particles. Every iteration, the velocity of each particle is updated. In the constriction-factor variant the velocity is updated according to [Clerc and Kennedy, 2002]:

$$\begin{aligned} \boldsymbol{v}_{i+1} &= \chi * (\boldsymbol{v}_i + c_1 * r_1 * (\boldsymbol{p}_i - \boldsymbol{x}_i) + c_2 * r_2 * (\boldsymbol{g} - \boldsymbol{x}_i)) \\ \chi &= \frac{2}{\left|2 - \phi - \sqrt{\phi^2 - 4\phi}\right|} \\ \phi &= c_1 + c_2 \end{aligned} \tag{3.7}$$

where:

$\boldsymbol{v}$ = vector of particle velocity
i = iteration number
c_1 = self confidence parameter
c_2 = swarm confidence parameter
$\boldsymbol{p}$ = location of personal best
$\boldsymbol{g}$ = location of global best
r = random number between 0 and 1

This function ensures that the updated velocity is a function of the original velocity, the direction towards the personal best, and the direction towards the global best. The ratio between their influence is dependent on some randomness as well as two parameters of the method: c_1 and c_2. They respectively represent the confidence in the personal best of each respective particle and the global best. The values of these parameters determine the emphasis of the algorithm, in which a higher c_1 value promotes more local search, and a higher c_2 value promotes more global search.

After updating the velocities of particles, their positions are are updated simply according to:

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \boldsymbol{v}_{i+1} \tag{3.8}$$

Until the maximum number of iterations has been met, this cycle continues starting with updating the personal and global best positions again.

3.3.3 Multi-Species Particle Swarm Optimization

The Multi-Species Particle Swarm Optimization (MSPSO) is an extension to the original PSO method, proposed by Iwamatsu. The extension adds a new speciation phase which splits up the swarm into multiple species, with the goal of locating multiple minima in the parameter space [Iwamatsu, 2006]. Figure 3.8 shows how the speciation phase is integrated with the original algorithm.

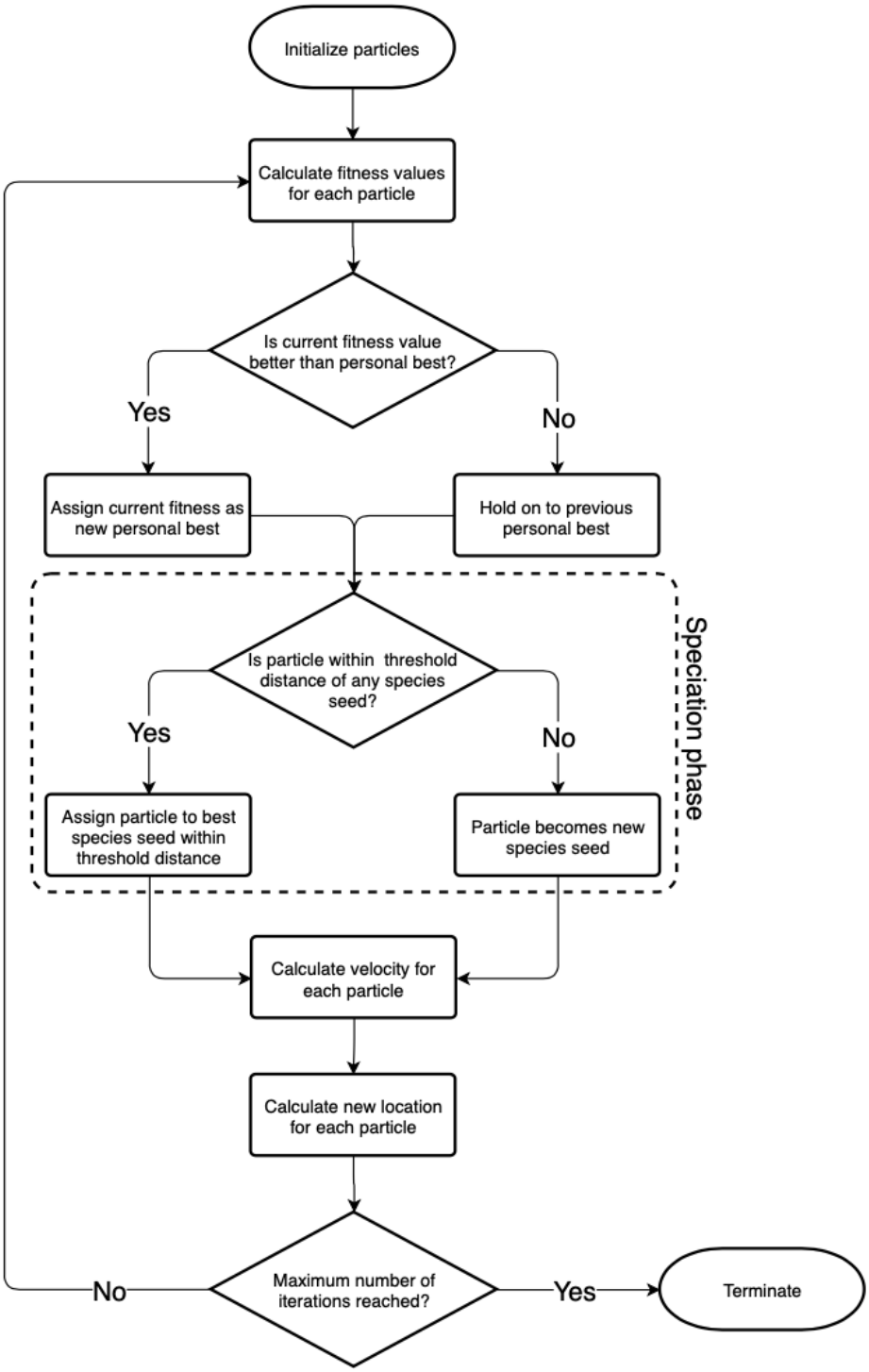


Figure 3.8: Diagram illustrating the multi-species particle swarm optimization workflow

The additional speciation phase consists of going over each particle in the swarm and checking whether it is within a certain threshold distance σ of any species seed. If this is not the case the particle itself becomes the seed for a new species. If this is the case, the particle becomes part of the species created by the seed. If a particle is within the threshold distance of multiple species seeds, the seed with the highest fitness value is chosen. This way seeds effectively create a hyper-sphere around themselves, within which particles can become part of that species. Figure 3.9 shows an example of the result of speciation for a 2-dimensional parameter space.

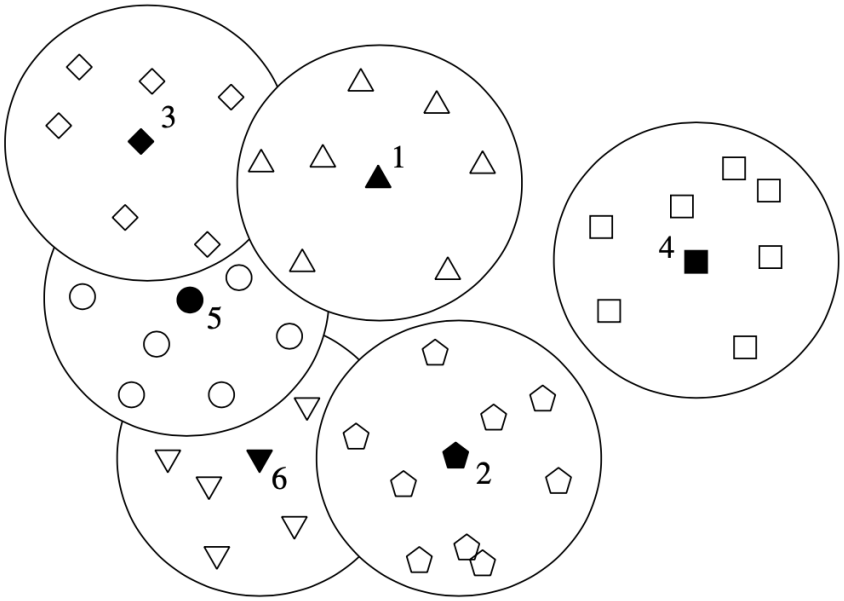


Figure 3.9: 2-dimensional spheres around species seeds determine which species each particle belongs to

Outside of the speciation phase the method functions mostly the same as the original PSO method does. To separate the global search of each species however, the global best variable in the velocity update function is replaced with the species best variable such that it becomes:

$$\boldsymbol{v}_{i+1} = \chi * (\boldsymbol{v}_i + c_1 * r_1 * (\boldsymbol{p}_i - \boldsymbol{x}_i) + c_2 * r_2 * (\boldsymbol{s}_i - \boldsymbol{x}_i)) \tag{3.9}$$

where:

$\boldsymbol{s}$ = location of species best

Unlike the base PSO algorithm or the ABC algorithm, the MSPSO algorithm inherently identifies local minima, and thus possible design alternatives through the species seeds. This could make the MSPSO algorithm very suitable for the proposed hybrid scheme in which the identification of multiple design alternatives is a primary objective.

3.4 Topology identification

Because the PSO and ABC algorithm posses no inherent mechanism to identify local minima, an additional step is required to extract from all the different solutions encountered during optimization, the ones which are worth showing to the end-user. For this purpose, an original method is proposed. The goal of the method is to find a variety of distinct structures of which the volume is close to that of the global minimum. It is therefore necessary for the method to be able to be able to distinguish slight variations in the structure from structures that we would consider to be distinct. The decision has been made to base the distinction between variations of structures and distinct structures on whether the change is purely geometrical (i.e. nodal locations) or topological as well (i.e. nodal connectivity). The goal is thus to be able to create a routine that is able to identify the topology of any structure. Let us consider the structure shown in Figure 3.10. Although the example structure is in 2D, the routine remains identical for 3D structures.

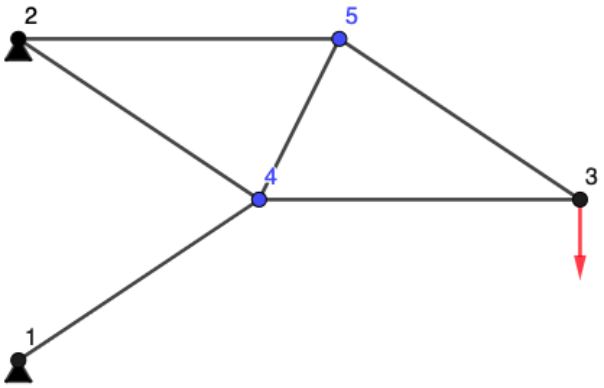


Figure 3.10: 2D example structure

A connectivity array is created by examining which of all possible members are present, and which are not. If there is a consistent method in which these members are allocated to indices of the array, then this array can be represented as a binary number which serves as an identifier for the topology of the structure. When this is done for the example structure according to the member ordering method shown in Table 3.1, the binary ID is equal to 0010011111. Since it is unpractical to store and compare binary representations of numbers, the binary ID is converted to a decimal ID, which in this case would be equal to 159.

Digit	1	2	3	4	5	6	7	8	9	10
Start node	1	1	1	1	2	2	2	3	3	4
End node	2	3	4	5	3	4	5	4	5	5
On or Off	0	0	1	0	0	1	1	1	1	1

Table 3.1: Example binary ID

This method of identifying distinct topologies is dependent on a constant ordering of the (movable) nodes. In this case an ordering from left to right is assumed. If this ordering would not be present then the method described above would identify a structure in which node 4 and 5 in Figure 3.10 are switched as distinct, while in reality they are equal. A common phenomenon that could disrupt the function of the topology identification method is when a structure does not contain all the available nodes (Figure 3.11).

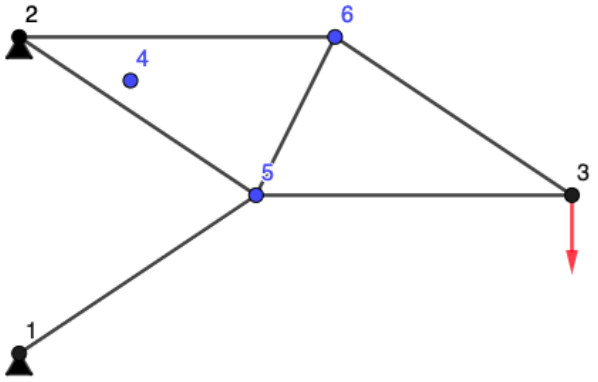


Figure 3.11: 2D example structure with one unconnected node

If no modification is made to the method, then the the binary ID of the structure would be equal to 000100011011001, and its decimal ID equal to 2265. However if the digits associated

with connections to the unconnected node are removed from the binary ID (Table 3.2), then the result is identical to the structure without the unconnected node.

Digit	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Start node	1	1	1	1	1	2	2	2	2	3	3	3	4	4	5
End node	2	3	4	5	6	3	4	5	6	4	5	6	5	6	6
On or Off	0	0	0	1	0	0	0	1	1	0	1	1	0	0	1

Table 3.2: Example binary ID with one unconnected node

One might point out that the decimal value of a binary number is not unique since adding any number of zero's to the lefthand side of the binary number results in an equal decimal value. The only way in which more digits are added to the binary number is if an additional node is connected to the structure (since unconnected nodes do not influence the ID). More specifically if an additional node is connected to a structure of n nodes, n more connections are possible and thus n digits are added to the binary number. For the decimal value of the binary number of the new structure to be equal to that of the original structure the leftmost n digits thus need to be equal to zero. These digits are associated with the members connected to the first node, and thus if these are zero, the first node is unconnected to the rest of the structure. This contradicts with the assumption that an additional node is connected to the structure and thus it is impossible for two binary connectivity arrays with an unequal number of digits to have an equal decimal value.

Unfortunately there are still some phenomena which cause this method to misidentify topologies: if two nodes are displaced such that their order based on the nodal coordinates is flipped, the displaced structure is misidentified as a distinct topology. Figure 3.12 shows an example of how the structure in Figure 3.10, when the movable nodes are ordered from left to right, could undergo a geometrical change that alters the topology ID.

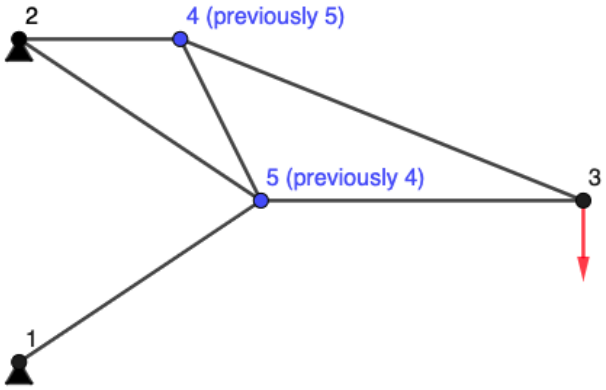


Figure 3.12: 2D example structure in which a geometry change causes a topological misidentification (movable nodes are ordered from left to right)

To catch any false negatives (when two topologies are equal but are labeled as distinct) by the ID method, an existing algorithm from graph theory is employed. Graphs are mathematical structures consisting of vertices connected by edges. It is therefore quite straightforward to represent a truss as an (unlabeled) graph. A set of unlabeled graphs of which their connectivity (i.e. topology) is identical is said to be isomorphic. The VF2 algorithm, originally developed by Cordella et al., is an efficient algorithm capable of conclusively testing whether a set of graphs is isomorphic or not [Cordella et al., 2001]. The VF2 algorithm has been implemented

in a graph-related Python package called NetworkX [Hagberg et al., 2008], which is used in the topology identification method. Although very efficient, the VF2 algorithm is still much more computationally expensive than generating a topology ID. Moreover, once a topology ID has been created for a structure it can be compared to any other structure with a topology ID almost instantly. Since the VF2 algorithm can only check a set of graphs, it is only employed to catch any false negatives from the ID method, as well as only for the structures which are candidates for being displayed to the end-user (more on this later).

Finally, there exists a difficulty in determining which elements are part of the structure, and which are not. The GSM determines the cross-sectional areas of all possible members in the structure via gradient-based optimization. This means that the members of which the cross-sectional area should be equal to zero are actually equal to some very small value. To determine which members are actually present in the structure, and which members only seem to be due to this numerical artifact, use is made of a method to distinguish members which are critical for the structure, and members which are not. A very simple method would be to employ a threshold value, defined as a percentage of the value of the largest cross-sectional area present in the structure. Any cross-sectional area below this threshold is then considered not to be present in the structure. If a relatively low threshold percentage is chosen (0.1%-1%), no important structural members are left out of the final structure. However, some structural members of lower importance remain in the final structures as well. This in itself is not an issue, however any alteration in the topology of these secondary members results in a different topology ID, even though the primary structures are identical. An example of such a case is shown in figure 3.13. The secondary members are barely visible due to their small cross-sectional areas, Figure 3.14 shows the same figures in which the secondary members are highlighted in green.

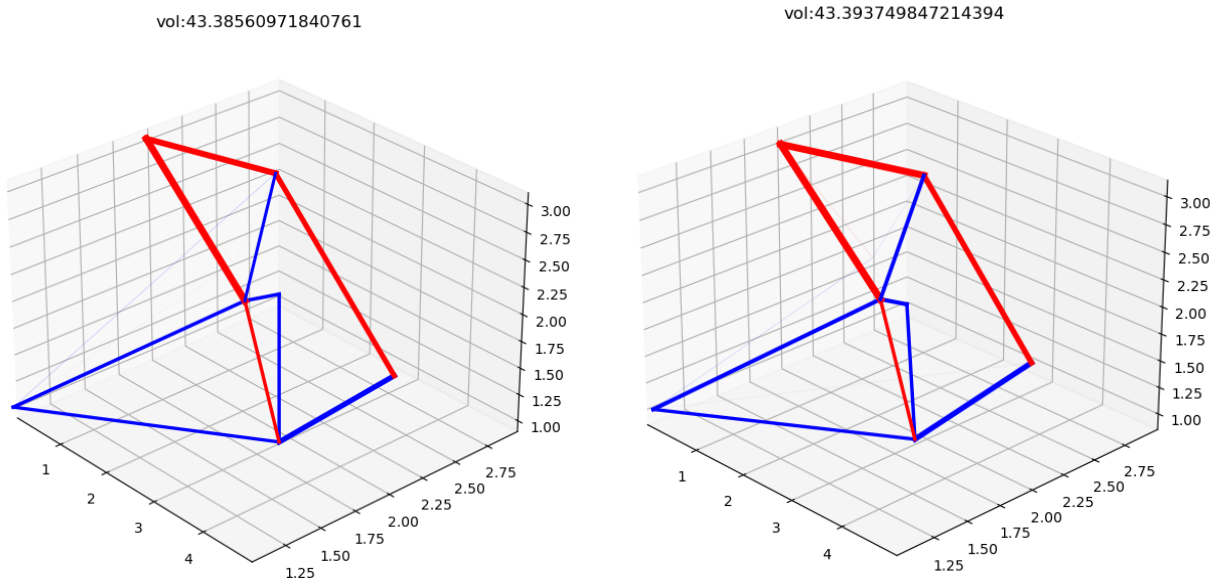


Figure 3.13: Two variants the same main topology, with differing topology in secondary members

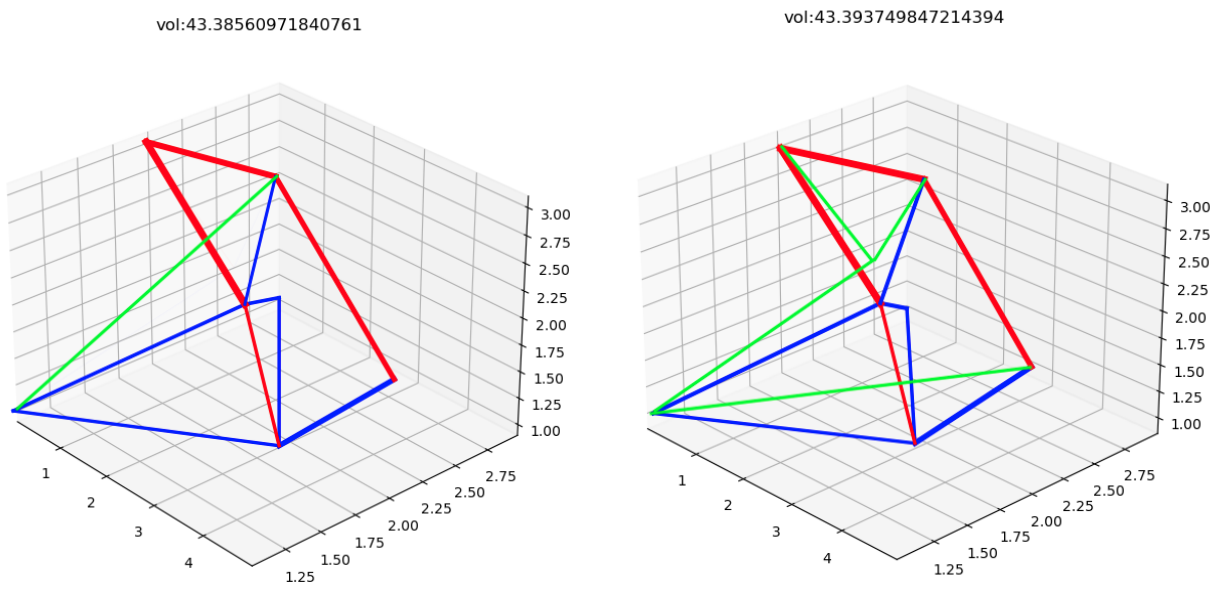


Figure 3.14: Two variants the same main topology, with differing topology in secondary members. Secondary members are highlighted in green

A solution to this problem could be to increase the threshold percentage. However if large threshold percentages are chosen (1%-10%), a new problem arises in which members that do have a significant contribution to the load carrying capacity can disappear. This results in strange and undesirable structures in which for example a node could be connected to the rest of the structure by only a single member (Figure 3.15).

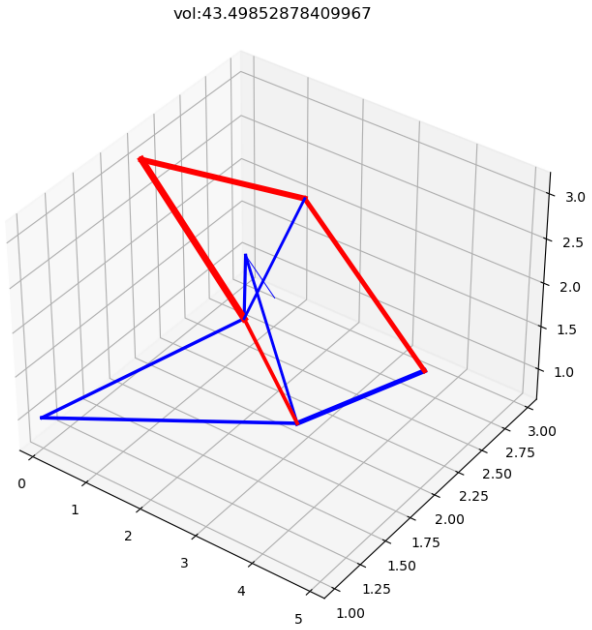


Figure 3.15: Example of a structure in which a node is connected to the structure by just one member

A more elaborate set of conditions is therefore employed to separate critical from non-critical members: firstly non-critical nodes are identified. Non-critical nodes are defined as nodes of which the connecting member with the largest cross-section area is below 30% of the nodal average of the largest connected cross-section area. This step effectively removes nodes, and

their connected members, to which no critical members are connected. Secondly, non-critical members are identified by the condition that their member area must be below 50% of the nodal average member area of its connecting nodes at both ends. In this way members which are deemed non-critical to both of its end-nodes are removed from the structure as well. Thirdly, nodes which are connected only to two other nodes are replaced with a single member directly connecting those two neighbouring nodes. While such nodes can not exist in the original structure, the removal of a non-critical member (in step 2) from a node connected to just three members will create a part of the structure where two members connected only to each other at one end effectively create a kinked member. Replacing this kinked member-construction with a single member avoids topological misidentification. Next, it is verified that the smallest distance between any two nodes is larger than 20% of the average distance between nodes. This step serves to remove structures of which many topological alternatives are possible without much change to the actual structure. Namely if two nodes are extremely close to one another, a member could switch its connectivity from one node to the other and thus changing the topology while the structure remains nearly unchanged.

This entire procedure is computationally quite costly and therefore unfeasible to execute for every solution that is found by the meta-heuristic algorithm. Instead, similarly to the VF2 algorithm, it is only employed for structures which are candidates for being displayed to the end-user. The total structure of the topology identification method consists effectively of two phases: the first phase simply consists of generating a topology ID for each solution that is generated during the execution of the meta-heuristic algorithm. The second phase takes place after the meta-heuristic algorithm has finished and has as its purpose to display a specified number of distinct high-quality solutions to the end-user. Figure 3.16 shows a diagram explaining the workflow of this second phase.

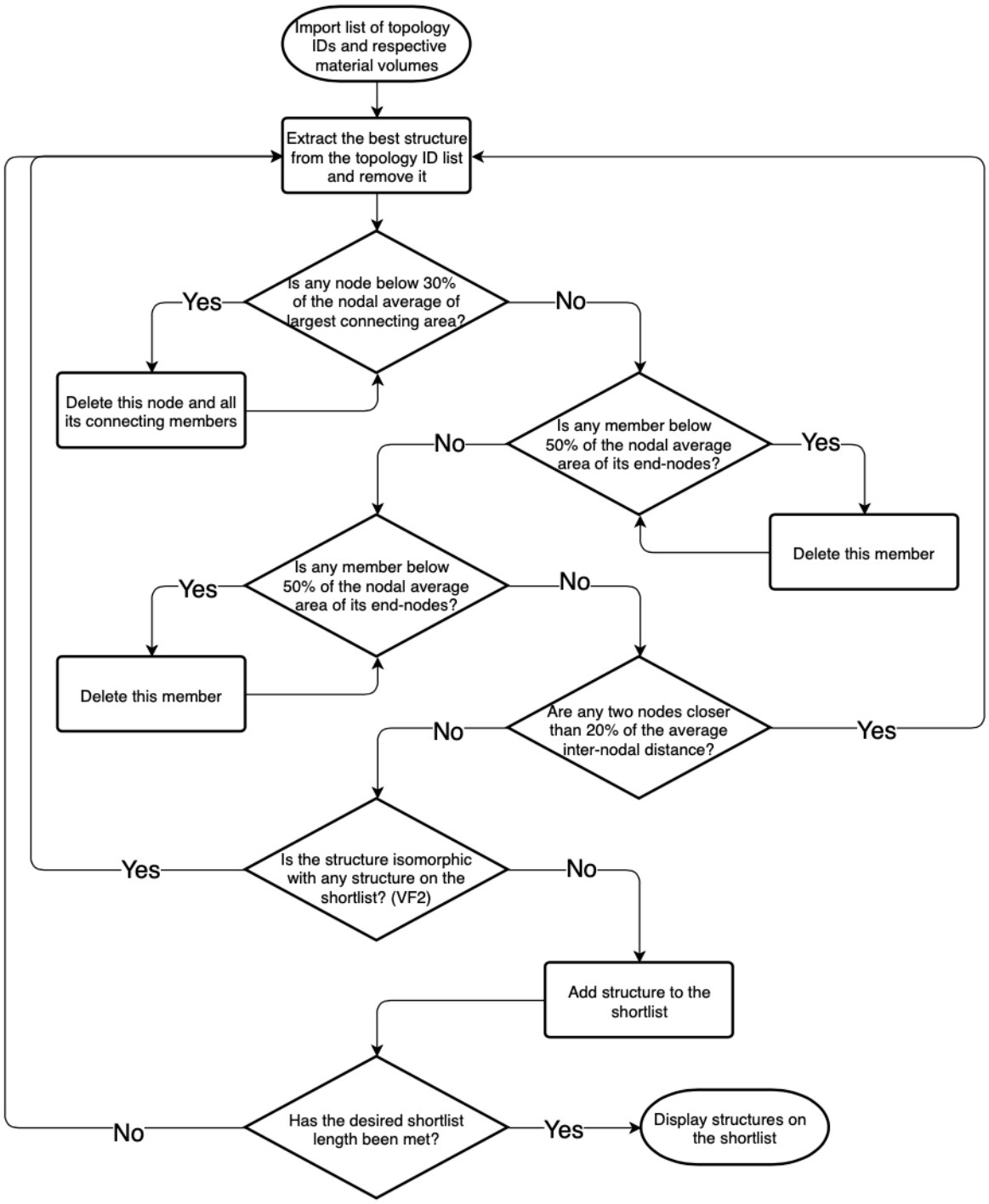


Figure 3.16: Diagram illustrating the workflow of phase 2 of the topology identification method

Chapter 4

Results

In this chapter each of the three hybrid method variants: ABC, PSO and MSPSO will be subjected to three case study problems. The results are presented and observations are made. While observed phenomena are explained, discussion of the results is left to chapter 5.

4.1 Case study problems

The first of the three case study problems is a 4-point cantilever, displayed in Figure 4.1. Three nodes, acting as supports, have fixed displacements in all three directions. A single point load acts downwards on the remaining unsupported node. These 4 nodes are fixed in location during the optimization process. Besides these 4 nodes, 4 additional movable nodes are present within the design space: a 5 by 4 by 4 rectangular cuboid. The boundaries of the design space have been chosen large enough such that they do not inhibit any potential optimal solutions. With 4 fixed nodes and 4 movable nodes, potential solutions can therefore contain at most 8 nodes. From hereon this problem will be referred to as problem 1a.

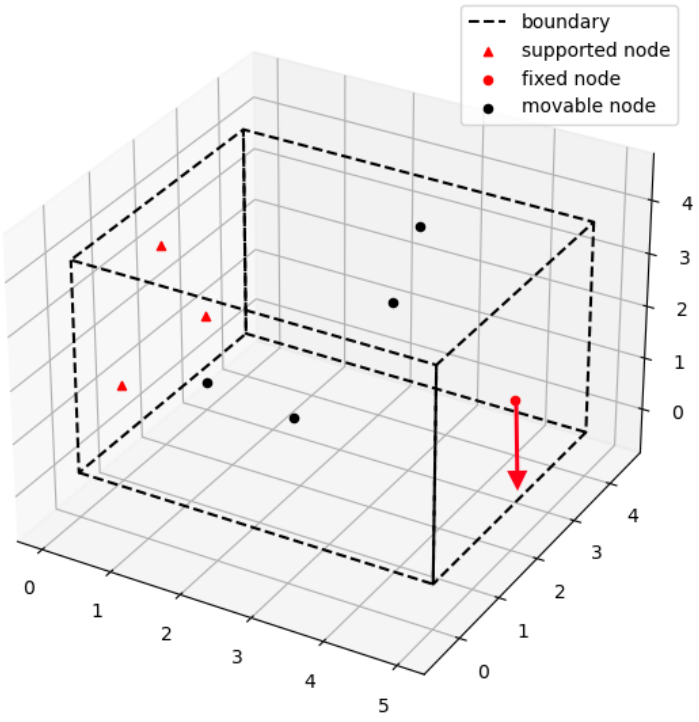


Figure 4.1: Case study problem 1a

The second case study (Figure 4.2) is a variant of the first. It is identical, except that the number of movable nodes has been increased from 4 to 8. This in turn increases the limit of nodes for potential solutions from 8 to 12. This case study is designed to compare the differences between the different method variants at different problem scales, while keeping most of the other variables the same. This problem will be referred to as problem 1b.

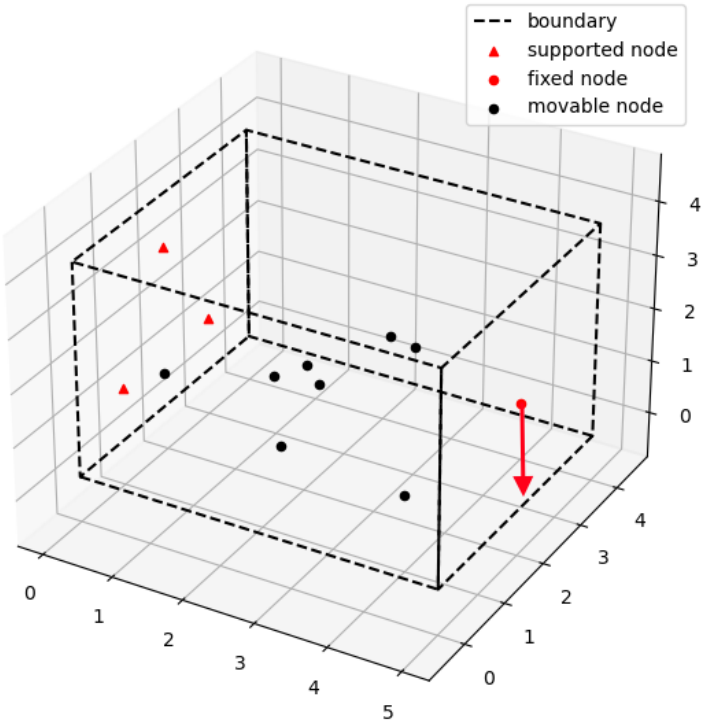


Figure 4.2: Case study problem 1b

The third and final case still resembles a cantilever, however this structure is subject to multiple load cases (Figure 4.3). It is emphasized that these loads resemble different load cases meaning that while the structure must be capable of resisting both loads, it is not required to resist both loads simultaneously. This case study is designed to be similar to case 1a but still contain multiple load cases. A comparison of both cases could therefore reveal the effect of the problem containing multiple load cases on the solution.

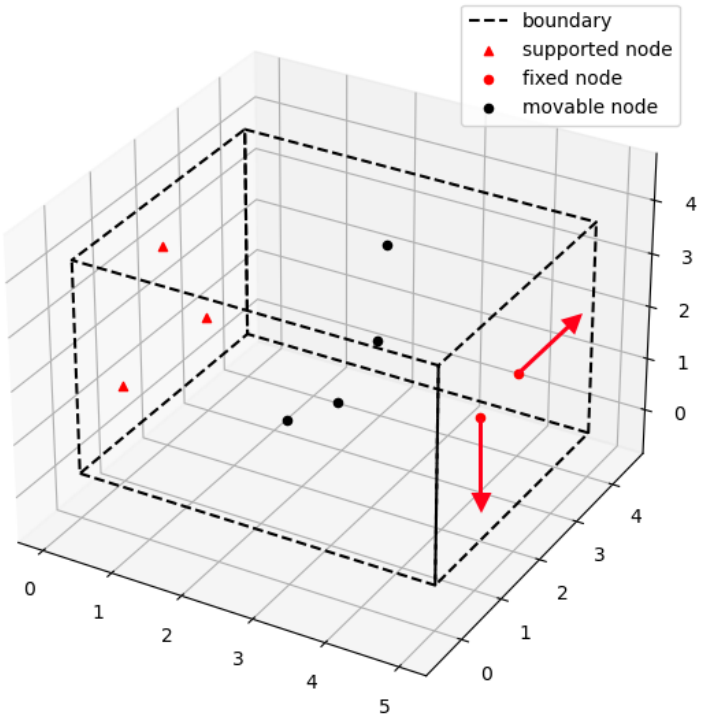


Figure 4.3: Case study problem 2

To enable a fair comparison between the three method variants, method parameters are set equal when they apply to multiple methods. Method parameters unique to any given method are set to appropriate values for the given problems. Table 4.1 gives an overview of the parameters used for each method variant.

Table 4.1: Overview of method parameter values

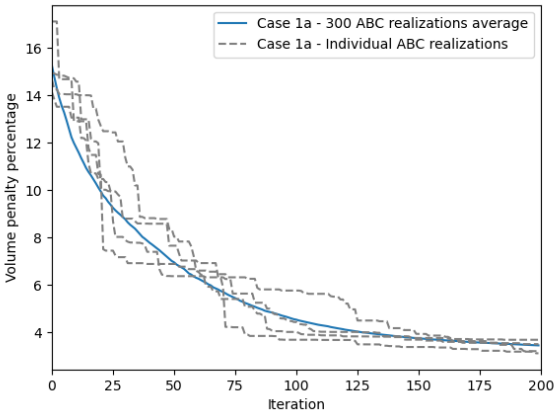
Parameter	ABC	PSO	MSPSO
Population	30	30	30
Max number of iterations	200[a]	200[a]	200[a]
Evaluation limit	100	N/A	N/A
$c1$	N/A	2.05	2.05
$c2$	N/A	2.05	2.05
Distance threshold	N/A	N/A	5.5[b]

[a] For case 1b 400 iterations are used, [b] for case 1b this value is set to 10

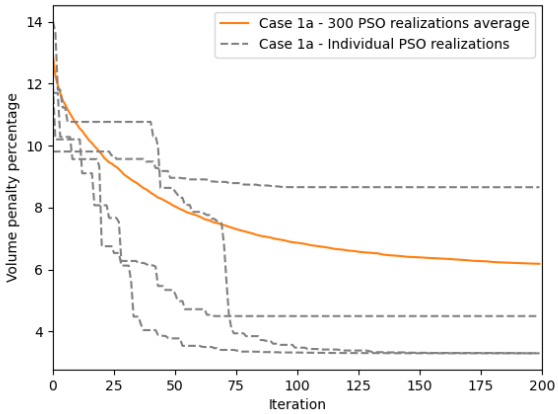
Quality of solutions will be measured in "volume penalty", this value is the percentual increase in volume as compared to a standard GSM solution with a dense nodal grid. Appendix A gives an overview of the derivation of these benchmarks. It should be noted that the benchmarks contain an extremely large number of members and nodes, while the hybrid method solutions are limited in their complexity. A volume penalty of zero is therefore virtually always unattainable. This means that even though a structure is labeled with a certain volume penalty, this does not mean that it could not represent the theoretical optimum, given its complexity constraints.

4.2 Convergence speed

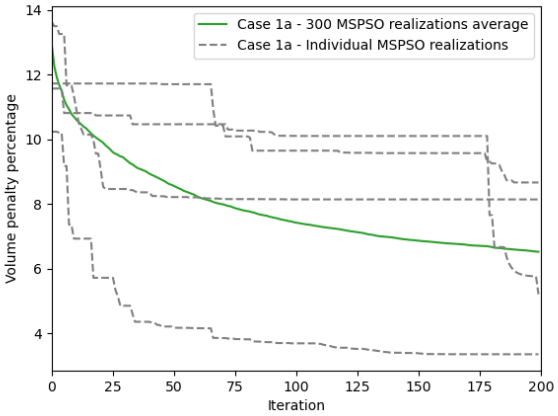
Figure 4.4 shows the convergence of the three method variants for case 1a. The vertical axis displays the volume penalty percentage of the least volume structure found by the method up until the iteration displayed on the horizontal axis. The average convergence of 300 independent realizations is shown, as is a sample of individual realizations for each method variant.



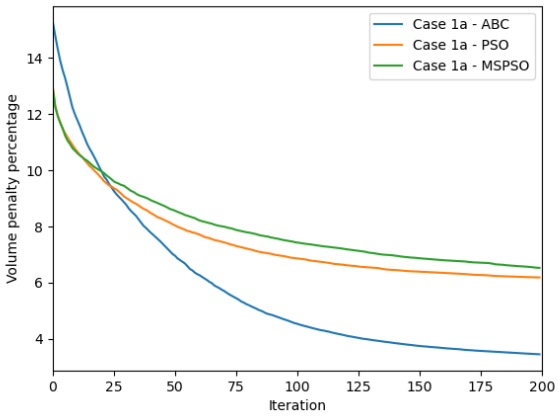
(a) ABC



(b) PSO



(c) MSPSO



(d) 300 realization averages

Figure 4.4: Individual realizations and 300 realization averages of convergence of ABC, PSO and MSPSO hybrid method variants for case study 1a

From figure 4.4d it is apparent that out of the three method variants, the ABC hybrid method convergences by far the fastest. The MSPSO variant converges the slowest, however its difference with the PSO variant is small compared to the difference between the PSO and ABC variants. Because the PSO and MSPSO method both start with a randomly initialized population of structures, it is expected that before the first iteration they have an equal average volume penalty. The reason that the ABC method initially has a higher average volume penalty, is that only half of its population (the employed bees) initializes a random structure. Thus while the PSO and MSPSO variants start with a volume penalty that is the lowest out of 30 random structures, the ABC variant starts with a volume penalty that is the lowest out of 15 random structures, which is naturally higher.

The samples of individual realizations indicate some differences between the three variants as well. The ABC variant not only converges the fastest to a high-quality solution, but it also

appears to be the most consistent in doing so. The individual realizations of the ABC method remain quite close to the 300 realization average for the entire optimization process. For the PSO and MSPSO method individual realizations deviate quite starkly from the 300 realization average with what seems to be, due to the long flat portions of the graphs, a tendency to get stuck in local minima. Figure 4.4c does however include one realization with a sharp drop-off in volume penalty succeeding a long period of it being constant. This could be an indication that given enough iterations, the PSO and MSPSO methods are capable of escaping local minima in which they have been stuck for some time.

The convergences of case study 1b and 2 have been studied as well, and their graphs display the same pattern as seen in Figure 4.4 to a great extent. The reader is referred to Appendix B for these graphs if further verification of the observations done in this section is desired.

4.3 Topological variety

The most insightful way of comparing topological variety between the methods is by examining the results produced by single realizations of the method variants. Effort has been made to select realizations which are representative for a typical result of the respective method variant. For all methods and case studies the 20 best design alternatives are displayed, with the exception of the MSPSO variant. Due to the nature of this method, and the somewhat limited population of 30 individuals, this method produces less than 20 design alternatives, of which all are displayed.

The members in the structures displayed in this section are coloured according to the following rules: members which are in tension for all load cases are displayed as red. Members which are in compression for all load cases are displayed as blue. Finally, members which are in tension for some load cases, while compression in others are displayed as gray.

4.3.1 Case study 1a

The design alternatives produced by the ABC variant for case study 1a (Figure 4.5) are mostly very similar. At first sight some structures might seem to possess identical topologies. The topology identification has made this impossible however, and upon closer inspection a difference in topology between any 2 displayed graphs, however small, is always found.

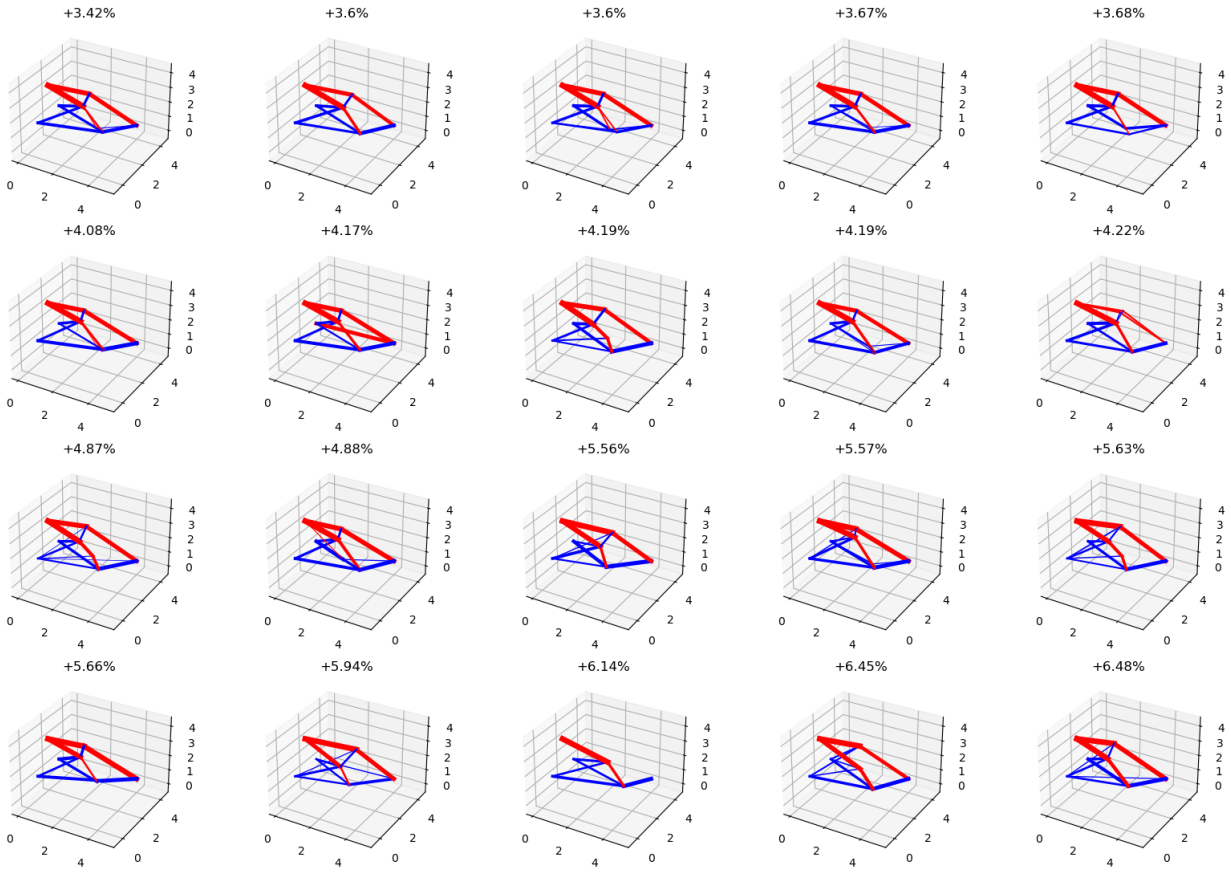


Figure 4.5: Design alternative variety of ABC hybrid method for case study 1a (Red = Tension; Blue = Compression)

While still quite monotonous, when compared to the results of the ABC variant, the PSO variant does demonstrate some more significant variety in its design alternatives. This does come at a slight cost in solution quality however (Figure 4.6). Where the ABC variant solutions range from 3.4% to 6.5% volume penalty, the PSO variant produces solutions with penalties ranging from 3.6% to 9.6%.

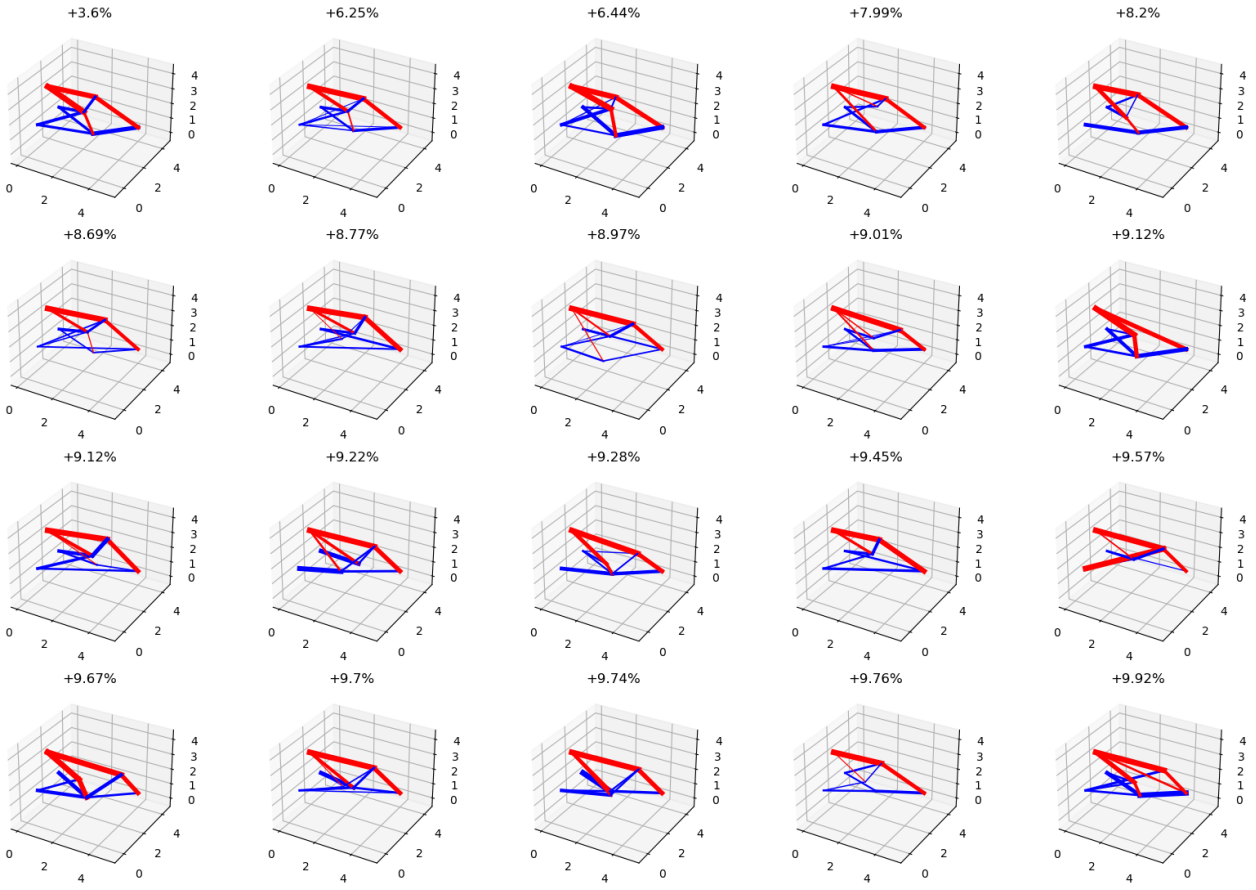


Figure 4.6: Design alternative variety of PSO hybrid method for case study 1a (Red = Tension; Blue = Compression)

Despite its low number of design alternatives, most of the alternatives displayed by the MSPSO variant are significantly distinct. The range of solution quality still suffers more however, going from 5.5% up to 14.2%.

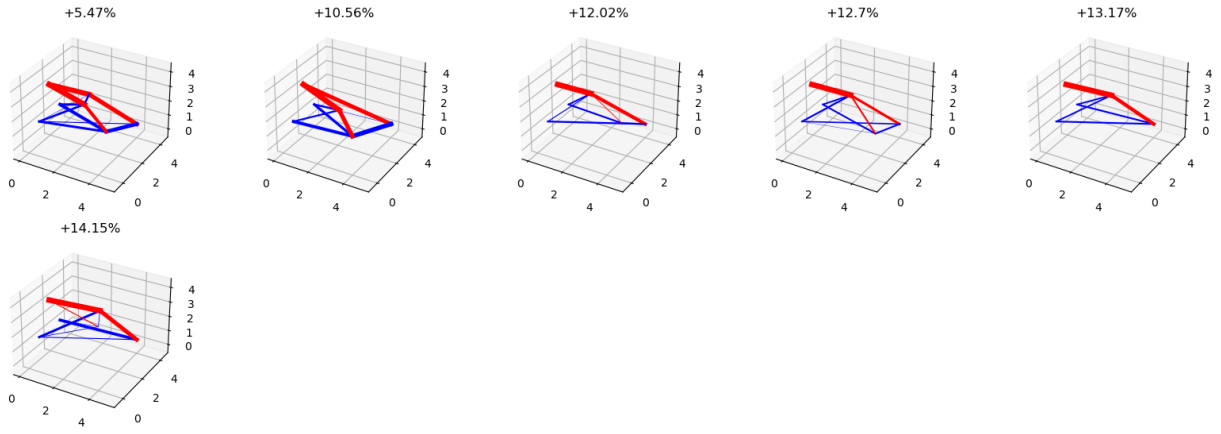


Figure 4.7: Design alternative variety of MSPSO hybrid method for case study 1a (Red = Tension; Blue = Compression)

4.3.2 Case study 1b

The volume penalties shown by the ABC design alternatives for case 1b (Figure 4.8) are significantly lower than those of case 1a. This reduction in volume penalty is due to the ABC variant incorporating the additional movable nodes at its disposal, in the structures. The result is a set of design of alternatives with lower volume penalties, but more complex structures containing more nodes and members as well. The topological variety remains similar to case 1a, where all variants are strictly topologically distinct, but their variations are small.

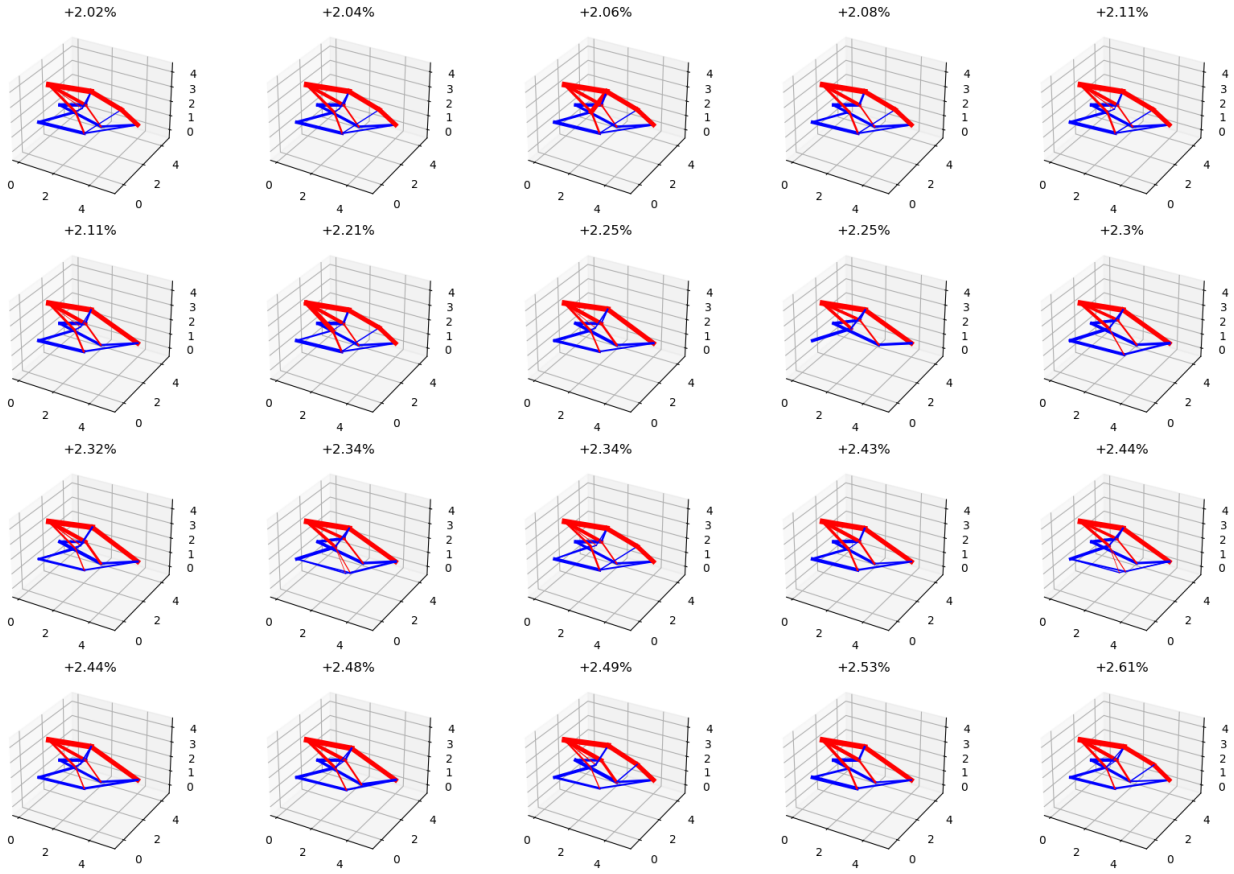


Figure 4.8: Design alternative variety of ABC hybrid method for case study 1b (Red = Tension; Blue = Compression)

Unlike the ABC variant, the PSO variant makes little use of the additional movable nodes of case 1b (Figure 4.9). Consequences are that the results are extremely similar to the PSO results of case 1a, both in topological variety as well as in solution quality.

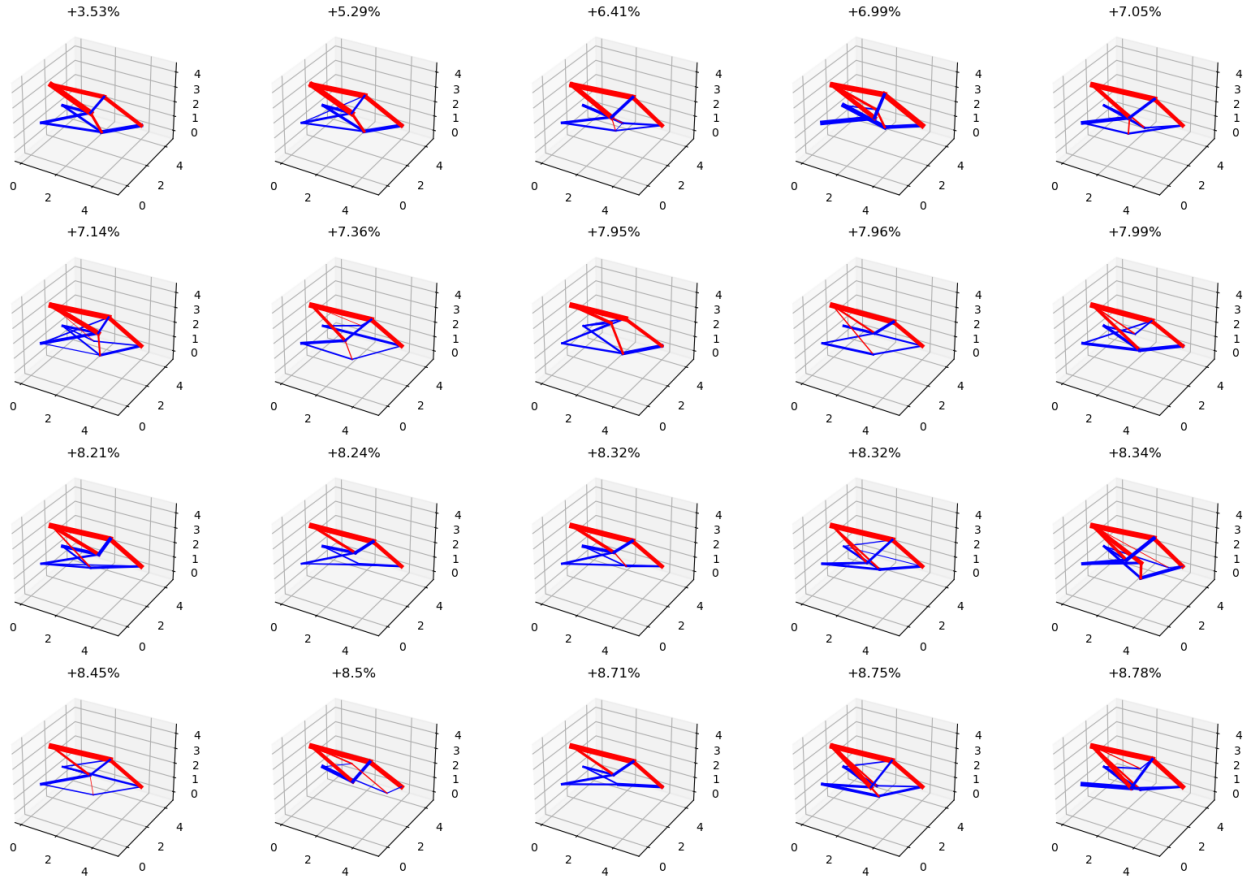


Figure 4.9: Design alternative variety of PSO hybrid method for case study 1b (Red = Tension; Blue = Compression)

Similarly to the PSO variant, the MSPSO variant is unable to effectively utilize the additional movable nodes introduced in this case study. The solutions of case 1b (Figure 4.10) retain their high topological variety and relatively low solution quality when compared to the results of case 1a.

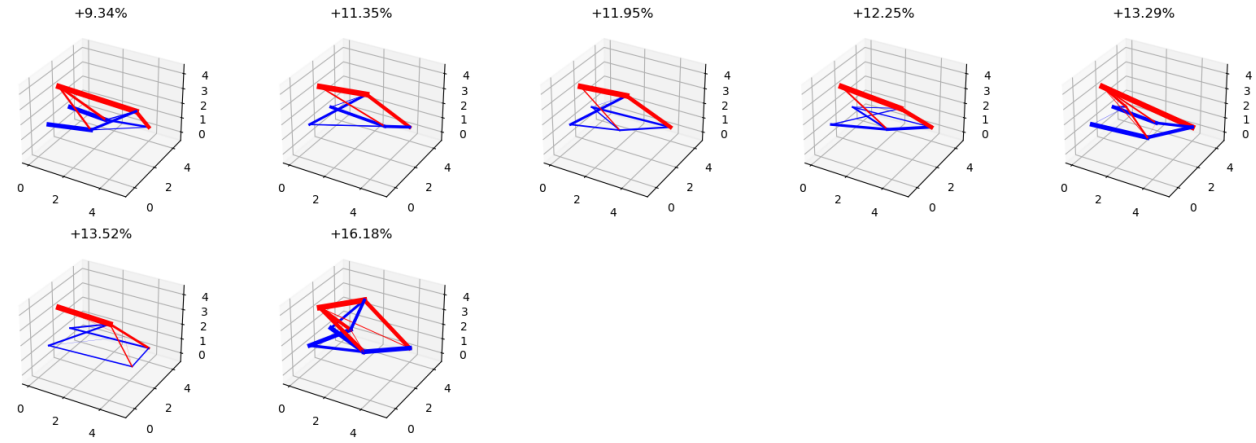


Figure 4.10: Design alternative variety of MSPSO hybrid method for case study 1b (Red = Tension; Blue = Compression)

4.3.3 Case study 2

Case study 2 introduces multiple load cases. Figure 4.11 displays the design alternatives produced by the ABC method variant. One could argue that the topological variety increases when compared to case study 1a, although at most its effect is small. What is apparent, is that despite a comparable number of nodes in the structure as case 1a, the structures produced for case 2 are significantly more complex with a higher degree of connectivity between the nodes.

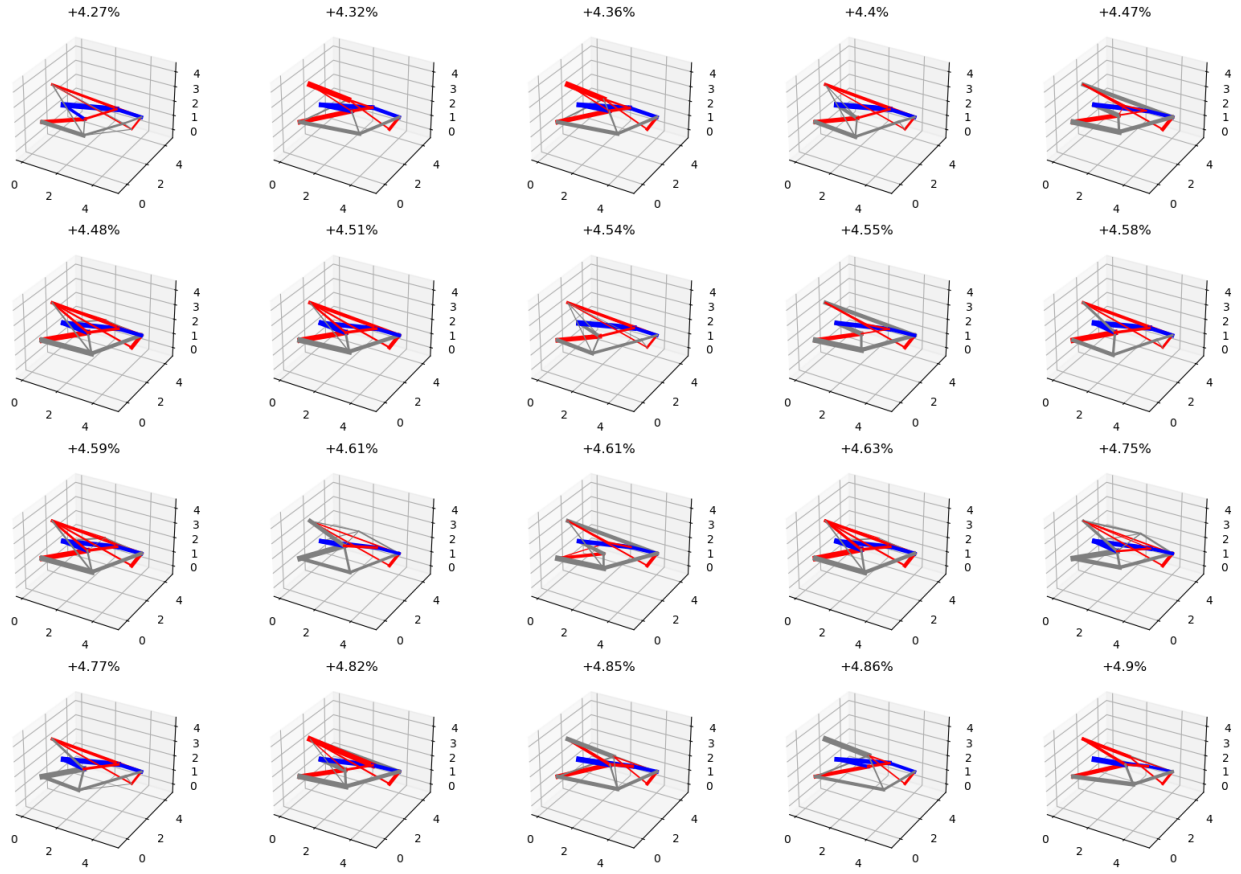


Figure 4.11: Design alternative variety of ABC hybrid method for case study 2 (Red = Tension; Blue = Compression; Grey = Tension and compression for different load cases)

Similarly to case 1a and b, the topological variety displayed by the PSO variant (Figure 4.12) can be said to be somewhat higher than that of the ABC variant. The PSO variant also has lower solution qualities once again, with a volume penalty that is roughly double that of the ABC variant solutions.

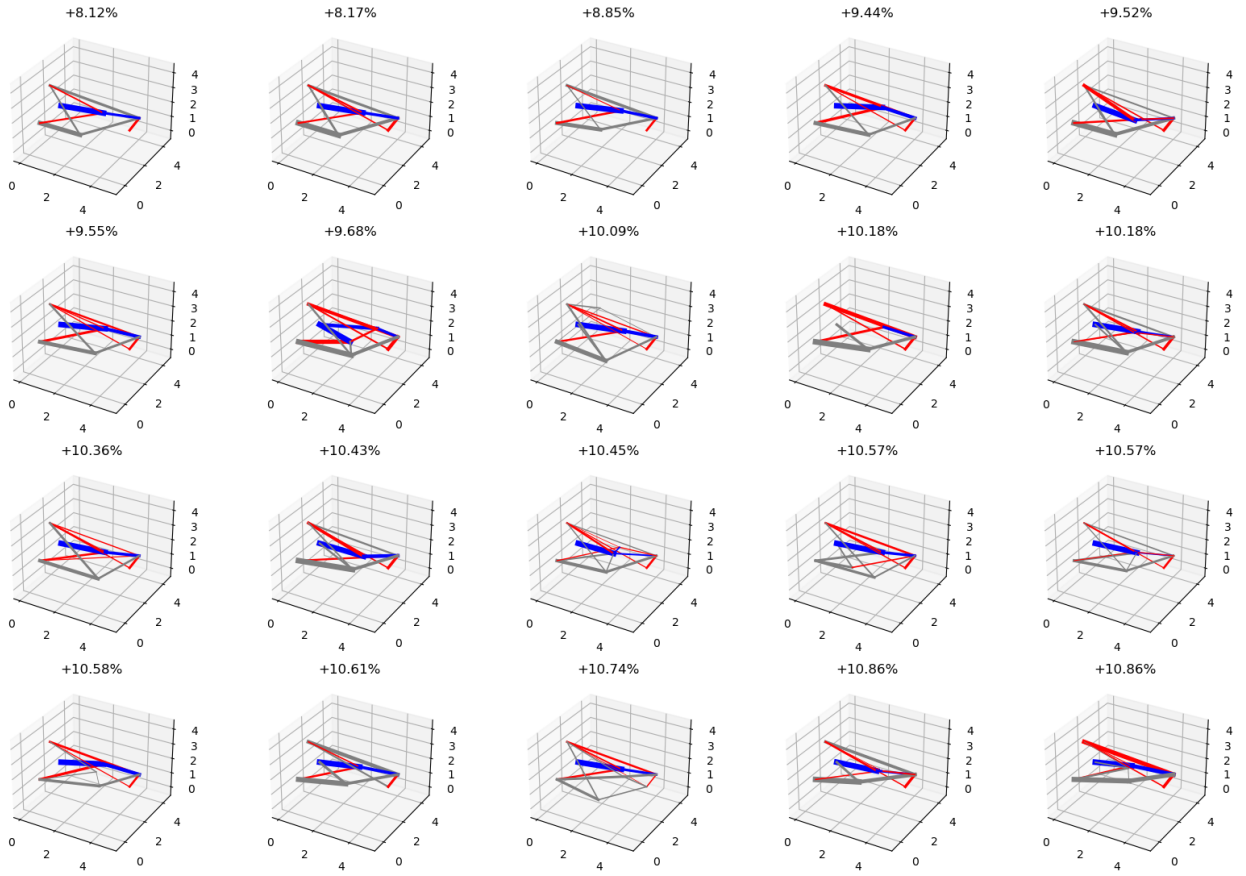


Figure 4.12: Design alternative variety of PSO hybrid method for case study 2 (Red = Tension; Blue = Compression; Grey = Tension and compression for different load cases)

Finally, the MSPSO variant produces once again design alternatives with the most topological variety (Figure 4.13).

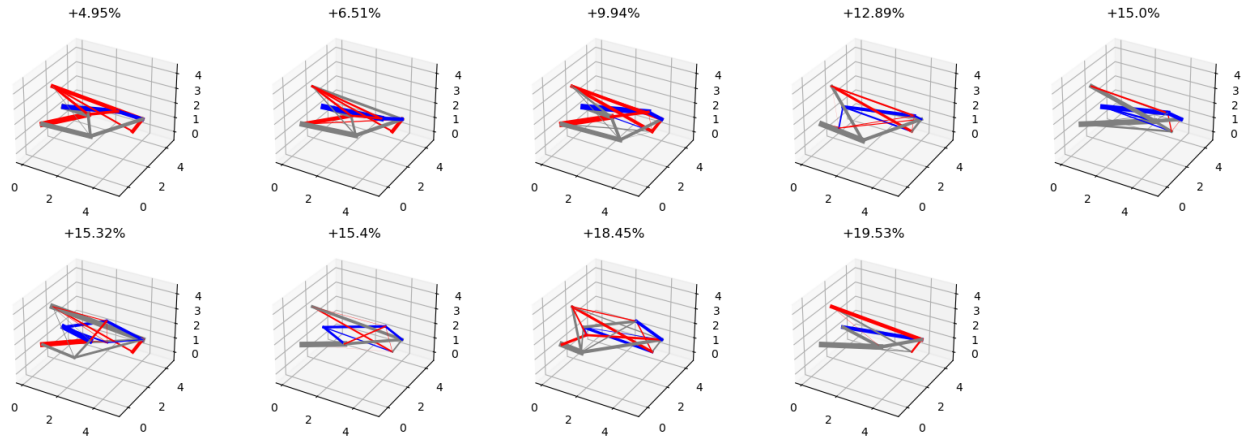


Figure 4.13: Design alternative variety of MSPSO hybrid method for case study 2 (Red = Tension; Blue = Compression; Grey = Tension and compression for different load cases)

4.4 Computational efficiency

4.4.1 Ground Structure Method optimizations

Figure 4.14 displays the GSM subroutine runtime for case 1a for different code implementations. The CVXPY API implementation is by far the slowest with a total runtime of 12.6 ms of which just 0.9 is spent in the solver. The remaining 11.7 ms is spent interpreting the problem formulation and converting to a form in which it can be solved efficiently. Using a basic MOSEK API implementation less than a third of this 11.7 ms conversion time is still left, with a conversion time of just 3.6 ms. By introducing the parametric scheme in which constants in the problem formulation are set up only once, the problem setup time is roughly cut in three once again, leaving 1.3 ms. Finally, making use of a sparse equilibrium matrix and implementing its sparsity pattern as a constant in the parametrized model, the setup time is reduced to 0.9 ms. In total a problem setup time speed up of 13 times is achieved. Due to the solver time being independent of implementation, the total GSM subroutine time is sped up by a factor of 7.

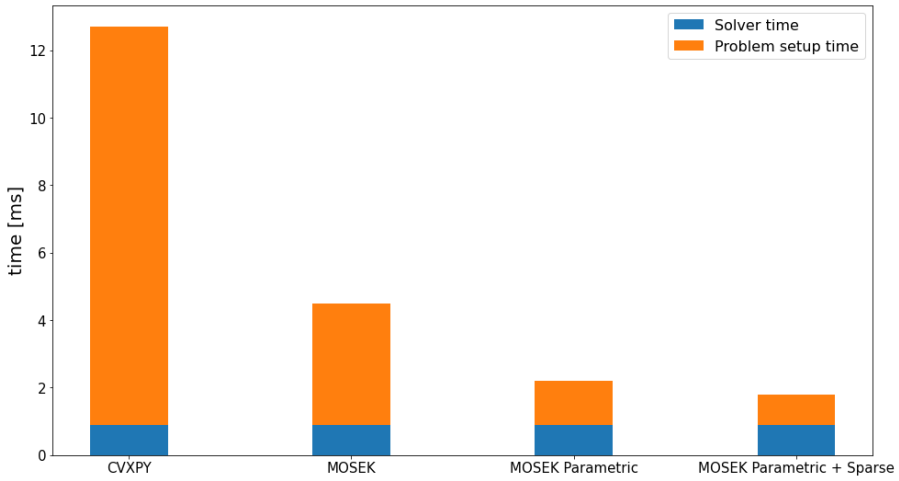


Figure 4.14: MOSEK API performance increase as compared to CVXPY API (case 1a)

4.4.2 Total iteration times

Next to the solver and problem setup time in the GSM subroutine, the third and final part of the algorithm taking up computation time is the logic within the meta-heuristic algorithms. Figure 4.15 displays the time spent in each of the three categories for each method variant, for each case study.

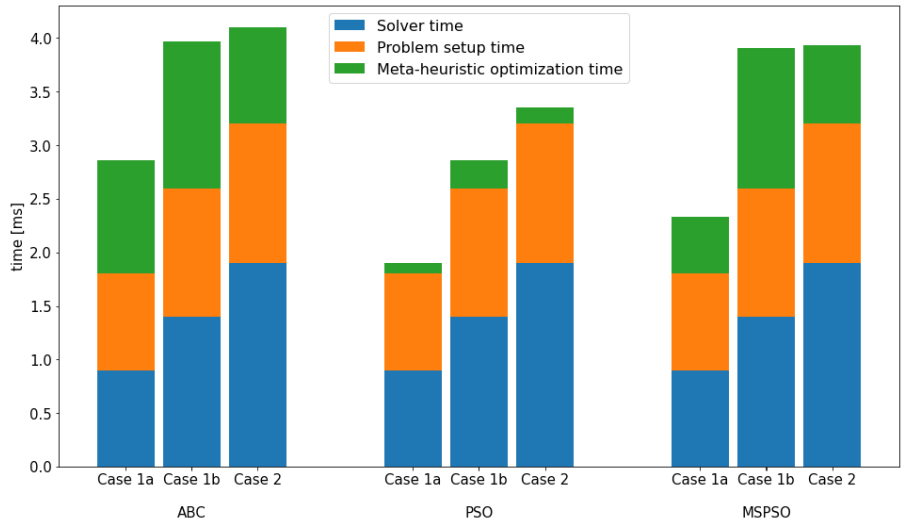


Figure 4.15: Computation time distribution comparison

A number of observations can be made: The solver time for case 1b is roughly 50% higher than that of case 1a. This increase in solver time is expected since the problem remains identical except for the addition of more (movable) nodes, and thus degrees of freedom to the GSM problem. The problem setup time increases as well, but only by approximately 33%. It is known that for large-scale problems the problem setup time becomes much less significant, which is consistent with this pattern of a more aggressive increase in solver time than problem setup time. Case 2 boasts an even larger solver, as well as problem setup time, despite containing 9 instead of 12 nodes. Apparently the addition of an extra load case is computationally more expensive than an increase from 9 to 12 nodes in the ground structure. The GSM problem setup time and solver time are equal for all three method variants. This is because all three variants use an identical GSM subroutine, which is the only factor in determining these computation times.

The meta-heuristic optimization does show significant differences in computation times between the method variants. With the ABC variant being the slowest and the MSPSO variant being only slightly faster, the PSO variant is much faster than its competitors. Because the PSO method is by far the simplest method of the three it requires the least amount of logic in its Python implementation. Since Python is a high-level programming language, differences in the amount of logic required can have a significant impact on the total computation time. Shifting the attention to the difference in meta-heuristic computation time between the case studies, it is apparent that case 1b is computationally the heaviest. This is to be expected since case 1b boasts double the number of movable nodes compared to case 1a and case 2. The meta-heuristic optimization time is independent of the number of load cases, and thus unlike for the solver and problem setup time, the meta-heuristic optimization time for case 2 is lower than that for case 1b.

When the comparatively small additional computation time of the topology ID method is taken into account, total run-times for the results displayed in this chapter range from 15 to 25 seconds, depending on the case and method variant. An exception to this are the results for case 1b: because of the doubling of the number of iterations from 200 to 400, results for these cases take roughly double the computation time to produce as well, ranging from 40 to 50 seconds.

Chapter 5

Discussion

5.1 Comparison of variants

Table 5.1 shows a ranking of the three method variants in the tested categories based on the results found in chapter 4.

Table 5.1: Ranking of method variants in the tested categories.

Category	ABC	PSO	MSPSO
Convergence speed	1	2	3
Topological variety	3	2	1
Computation time	3	1	2

The fast convergence speed of the ABC variant is a huge advantage compared to the other variants. It is likely however that it is also partially responsible for its observed low topological variety. The ABC variant finds a good solution so quickly because, due to its division of labor, it is able to focus its attention to the best solutions in the population. Because of this attention to the best solutions, many slight variants of this solution are generated. Since the topology ID method is unable to distinguish slightly topologically different structures from structures which are entirely distinct, the best n distinct structures consist of mostly slight variants of this one good solution. Because of how the topology ID method works, the PSO method, which is effectively worse at solving the geometry optimization problem than the ABC method is, results in more topological variety. The MSPSO variant produces results with the best topological variety. This is naturally due to it not making use of the topology identification method, but rather by its build-in mechanism to find multiple solutions with a predetermined minimum parameter space distance between them. Unfortunately due to the MSPSO method being a PSO method extension, its convergence to quality solutions is very similar to the PSO variant.

The fact that even the worst structure produced by the MSPSO method has a material penalty of less than 20% does reveal some characteristics about the hybrid scheme as well. It is not rare for structural engineers to design structures with volume penalties of more than 30%. The method might be unable to find these structures because only non-optimal geometries are tolerated. The topology of any given structure is always optimal due to the convex optimization of the modified GSM. Given the nodal locations of a Pratt truss for example, the GSM will generate a topology which does not resemble a Pratt truss. While high volume penalty structures such as the Pratt truss are not desired to be found by the method, it is likely that lower volume penalty structures with non-optimal topologies are out of reach for the hybrid scheme as well.

As for computation time, the PSO method is the fastest of the three, which does give it a genuine advantage over the ABC and MSPSO variants with their current prototype implementations. An implementation build for actual usage by engineers would change this however: due to the usage of a highly efficient solver package, which for the ABC and MSPSO variants takes up roughly a third of the total runtime, the solver time will not see any reduction for a more polished implementation of the methods. The problem setup time could possibly still be reduced, however due to the already implemented optimizations it is very likely that a significant part of its computation time will remain. The meta-heuristic optimization time could be made much faster than it is in the current state, implementing it in a lower level programming language, such as C, instead of Python. It is not unrealistic that speedups of tens, up to ten thousand times, are possible. Such speedups would make the meta-heuristic optimization time negligible compared to the problem setup and solver time. This would effectively make the speed advantage the PSO variant currently has over the ABC and MSPSO variants, vanish.

5.2 Practicality

The limitation on the number of nodes in the structure has, in most cases, the desired effect of limiting the number of members, connections and therefore complexity of the structure. An exception to this is the case in which multiple load cases are considered. For these structures the connectivity between the nodes is quite dense, resulting in more complex structures. While at least for the single load case problems the structures are limited in its number of members, the angles between members are highly irregular. The effect this has on the constructability of the structures is highly dependent on the available manufacturing techniques.

Computation times of 15 to 25 seconds for small-scale problems are, given enough variety in the design alternatives, acceptable. For most topology optimization methods it is desired to have any particular run take up to at most a couple of seconds. Such short computation time targets stem from the tendency of users to run slightly different versions of the same optimization consecutively. This way the user is exploring different design alternatives by changing parameters or boundary conditions. The proposed hybrid methods (intend to) produce a selection of design alternatives for every run. Thus larger computation times are not only understandable, but acceptable as well. For larger-scale problems, however, computation times quickly increase. A doubling of the number of movable nodes, and thus the number of degrees of freedom for the meta-heuristic algorithm (the GSM experiences an even larger increase in number degrees of freedom) resulted in a 2 to 3 times increase in total computation time. Because there remain only relatively small speed-ups to be gained in the GSM subroutine, the meta-heuristic algorithm will need to be optimized for the number of GSM calls to make larger-scale problems feasible as well.

Chapter 6

Conclusion

6.1 Research questions

The main research question is repeated:

"Is a method based on the proposed hybrid scheme capable of generating multiple good design alternatives?"

The answer to this question, for any of the implemented methods is debatable: the ABC variant of the hybrid scheme is capable of producing good results, however the variety in its design alternatives is limited. The MSPSO variant produces moderately low volume structures as well as exhibiting a reasonable variety of design alternatives. It is likely however that with a more advanced topology identification method for the ABC variant, or a faster convergence for the MSPSO variant, a set of good design alternatives is achievable.

The first of two sub-questions was as follows:

"Which of the considered meta-heuristic algorithms is best-suited for the proposed hybrid scheme?"

In the current state of the three methods, the MSPSO algorithm is considered to be the best-suited, due to its ability of finding the most distinct design alternatives. Due to the fast convergence of the ABC algorithm however, the author's judgement is that the ABC variant has the most potential.

"For what problem size can a method based on the proposed hybrid scheme produce results within a reasonable time-frame on a standard PC?"

Problems of similar size to the three case studies (3-dimensional problems with up to 12 nodes or 2 load cases) can be solved in as little as 15 seconds and always in less than one minute on most contemporary desktop PCs. Considering that each run produces multiple design alternatives, less iteration is needed by the end-user which renders these computation times as acceptable.

6.2 Future research

For future research that directly builds on the work presented in this thesis, it is recommended that a more advanced topology identification method is developed, with the goal of increasing the design alternative variety. Other markers than solely topology could possibly be used to

distinguish design alternatives. Borrowing from the MSPSO algorithm for example, the distance between alternatives in the parameter space could be a good indicator for distinguishing design alternatives. In any case it has become clear from this research that equating every possible topological change to a distinction in design alternatives leads to an insufficient design alternative variety.

Alternatively, artificial intelligence technologies could be used to determine whether structures are distinct without a clearly defined definition on when they are. If somehow a sufficiently sized data set of structures with accompanying information about how distinct they are from one another could be acquired or generated, a machine learning algorithm might be very effective for this purpose. Generation of such a data set could prove to be quite difficult however: structures would have to be generated of which it is known that they are considered to be distinct, however in order to generate these structures, the algorithm resulting from the machine learning process is needed. An alternative way of obtaining such a data set would be by having people directly determine whether a set of structures is distinct or not. Depending on the number of data required however, this task is likely to be at least equally challenging.

More fundamentally, the way in which the problem is set up could be reconsidered as well. More ways of creating a non-convex parameter space other than only by the separation of geometric and topological parameters as has been done in this research can be imagined. Other non-convex problem setups could possibly be more easy to explore, or less constricting on the designs which can be generated.

Bibliography

[Ab Wahab et al., 2015] Ab Wahab, M. N., Nefti-Meziani, S., and Atyabi, A. (2015). A comprehensive review of swarm optimization algorithms. *PLoS ONE*, 10(5):1–36.

[Bendsøe et al., 1994] Bendsøe, M. P., Ben-Tal, A., and Zowe, J. (1994). Optimization methods for truss geometry and topology design. *Structural Optimization*, 7(3):141–159.

[Bendsøe and Sigmund, 2003] Bendsøe, M. and Sigmund, O. (2003). *Topology Optimization: Theory, Methods and Applications*. Springer.

[Boyd and Vandenberghe, 2004] Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

[Christensen and Klarbring, 2009] Christensen, P. and Klarbring, A. (2009). *An Introduction to Structural Optimization*. Springer.

[Clerc and Kennedy, 2002] Clerc, M. and Kennedy, J. (2002). The Particle Swarm—Explosion, Stability, and Convergence in a Multidimensional Complex Space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73.

[Cordella et al., 2001] Cordella, L. P., Foggia, P., Sansone, C., and Vento, M. (2001). An improved algorithm for matching large graphs. *3rd IAPR-TC15 workshop on graph-based representations in pattern recognition*, pages 149–159.

[Deb, 2000] Deb, K. (2000). An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2):311 – 338.

[Diamond and Boyd, 2016] Diamond, S. and Boyd, S. (2016). CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17:1–5.

[Dorn et al., 1964] Dorn, W., Gomory, R., and Greenberg, H. (1964). Automatic design of optimal structures. *Journal de Mècanique*, 3:25–52.

[Fairclough and Gilbert, 2020] Fairclough, H. and Gilbert, M. (2020). Layout optimization of simplified trusses using mixed integer linear programming with runtime generation of constraints. *Structural and Multidisciplinary Optimization*, 61(5):1977–1999.

[Gilbert and Tyas, 2003] Gilbert, M. and Tyas, A. (2003). Layout optimization of large-scale pin-jointed frames. *Engineering Computations (Swansea, Wales)*, 20(7-8):1044–1064.

[Hagberg et al., 2008] Hagberg, A. A., Schult, D. A., and Swart, P. J. (2008). Exploring network structure, dynamics, and function using NetworkX. *7th Python in Science Conference (SciPy 2008)*, pages 11–15.

[He and Gilbert, 2015] He, L. and Gilbert, M. (2015). Rationalization of trusses generated via layout optimization. *Structural and Multidisciplinary Optimization*, 52(4):677–694.

[Holland, 1975] Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* MIT Press.

[Homaifar et al., 1994] Homaifar, A., Qi, C. X., and Lai, S. H. (1994). Constrained optimization via genetic algorithms. *Simulation*, 62(4):242–254.

[Hooshmand and Campbell, 2016] Hooshmand, A. and Campbell, M. I. (2016). Truss layout design and optimization using a generative synthesis approach. *Computers and Structures*, 163:1–28.

[Iwamatsu, 2006] Iwamatsu, M. (2006). Locating all the global minima using multi-species particle swarm optimizer: The inertia weight and the constriction factor variants. *2006 IEEE Congress on Evolutionary Computation, CEC 2006*, pages 816–822.

[J. Kennedy, 1995] J. Kennedy, R. E. (1995). Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, pages 1942–1948.

[Joines and Houck, 1994] Joines, J. A. and Houck, C. R. (1994). On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's. *IEEE Conference on Evolutionary Computation - Proceedings*, 2:579–584.

[Karaboga, 2005] Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. *Erciyes University.*

[Karaboga and Basturk, 2007] Karaboga, D. and Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, 39(3):459–471.

[L. He, 2019] L. He, M. Gilbert, X. S. (2019). A python script for adaptive layout optimization of trusses. *Structural and Multidisciplinary Optimization*, 60:835–847.

[Luh and Lin, 2011] Luh, G. C. and Lin, C. Y. (2011). Optimal design of truss-structures using particle swarm optimization. *Computers and Structures.*

[Maheri et al., 2016] Maheri, M. R., Askarian, M., and Shojaee, S. (2016). Size and topology optimization of trusses using hybrid genetic-particle swarm algorithms. *Iranian Journal of Science and Technology - Transactions of Civil Engineering*, 40(3):179–193.

[Martínez et al., 2007] Martínez, P., Martí, P., and Querin, O. M. (2007). Growth method for size, topology, and geometry optimization of truss structures. *Structural and Multidisciplinary Optimization*, 33(1):13–26.

[Michalewicz, 1996] Michalewicz, Z. (1996). Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation*, 4(1):1–32.

[Michalewicz and Janikow, 1996] Michalewicz, Z. and Janikow, C. Z. (1996). GENOCOP: A Genetic Algorithm for Numerical Optimization Problems with Linear Constraints. *Communications of the ACM*, 39:175.

[Michalewicz and Nazhiyath, 1995] Michalewicz, Z. and Nazhiyath, G. (1995). Genocop iii: a co-evolutionary algorithm for numerical optimization problems with nonlinear constraints. In *Proceedings of 1995 IEEE International Conference on Evolutionary Computation*, volume 2, pages 647–651 vol.2.

[Michell, 1904] Michell, A. (1904). The limits of economy of material in frame-structures. *Philosophical Magazine*, 8(47):589–597.

[Mueller, 2014] Mueller, C. T. (2014). *Computational Exploration of the Structural Design Space*. PhD thesis, Massachusetts Institute of Technology.

[Parkes, 1975] Parkes, E. W. (1975). Joints in optimum frameworks. *International Journal of Solids and Structures*, 11(9):1017–1022.

[Plevris and Papadrakakis, 2011] Plevris, V. and Papadrakakis, M. (2011). A Hybrid Particle Swarm-Gradient Algorithm for Global Structural Optimization. *Computer-Aided Civil and Infrastructure Engineering*, 26(1):48–68.

[Prager, 1977] Prager, W. (1977). Optimal layout of cantilever trusses. *Journal of Optimization Theory and Applications*, 23(1):111–117.

[Pritchard et al., 2005] Pritchard, T., Gilbert, M., and Tyas, A. (2005). Plastic layout optimization of large-scale frameworks subject to multiple load cases, member self-weight and with joint length penalties.

[Shea, 1997] Shea, K. (1997). *Essays of Discrete Structures: Purposeful Design of Grammatical Structures by Directed Stochastic Search*. PhD thesis, Carnegie Institute of Technology.

[Sokól, 2014] Sokól, T. (2014). Multi-load truss topology optimization using the adaptive ground structure approach. *Recent Advances in Computational Mechanics - Proceedings of the 20th International Conference on Computer Methods in Mechanics, CMM 2013*, pages 9–16.

[Sonmez, 2011] Sonmez, M. (2011). Artificial Bee Colony algorithm for optimization of truss structures. *Applied Soft Computing Journal*, 11(2):2406–2418.

[Stiny and Gips, 1972] Stiny, G. N. and Gips, J. (1972). Shape Grammars and the Generative Specification of Painting and Sculpture," Information Processing 71, IFIP, North-Holland, Amsterdam. *Information Processing*, 71:125–135.

[Storn and Price, 1997] Storn, R. and Price, K. (1997). Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11(4):341–359.

[Torii et al., 2016] Torii, A. J., Lopez, R. H., and Leandro, L. F. (2016). Design complexity control in truss optimization. *Structural and Multidisciplinary Optimization*, 54(2):289–299.

[Wang and Ohmori, 2010] Wang, H. and Ohmori, H. (2010). Truss optimization using genetic algorithm, considering construction process. *International Journal of Space Structures*, 25(4):205–215.

[Wolpert and Macready, 1997] Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82.

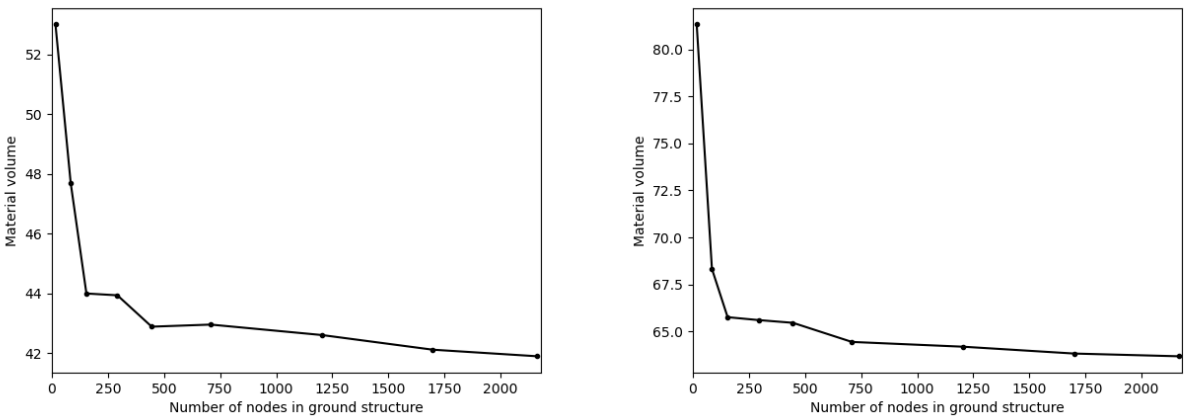
[Wu and Tseng, 2010] Wu, C. Y. and Tseng, K. Y. (2010). Truss structure optimization using adaptive multi-population differential evolution. *Structural and Multidisciplinary Optimization*, 42(4):575–590.

[Zuo et al., 2014] Zuo, W., Bai, J., and Li, B. (2014). A hybrid OC-GA approach for fast and global truss optimization with frequency constraints. *Applied Soft Computing Journal*, 14(PART C):528–535.

Appendix A

GSM Benchmarks

The benchmarks used to compute volume penalty values are obtained from large-scale GSM solutions. A study has been done in which for both case study problems convergence of the material volume has been analysed, as the density of the ground structure increases. The material volume corresponding to the ground structures with the most nodes at the right end of the graphs in Figure A.1 are used as benchmarks. Case studies 1a and 1b use the same benchmark since their only difference lies in the complexity constraints, which do not affect the method by which the benchmark is derived.



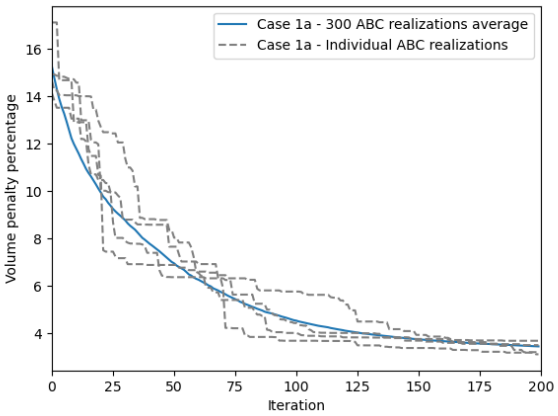
(a) Case study 1

(b) Case study 2

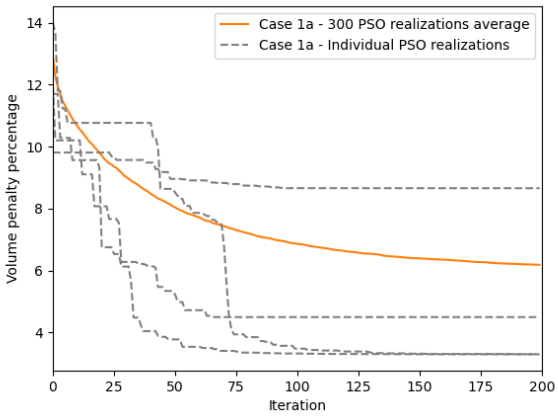
Figure A.1: Standard GSM solution material volume versus nodes in ground structure

Appendix B

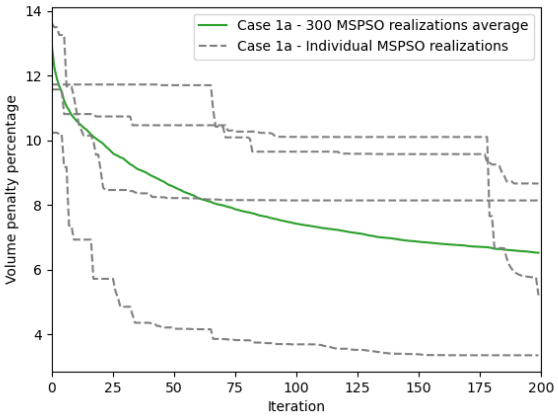
Extended convergence study



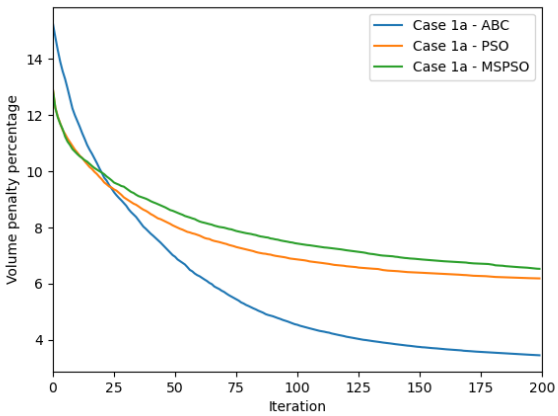
(a) ABC



(b) PSO

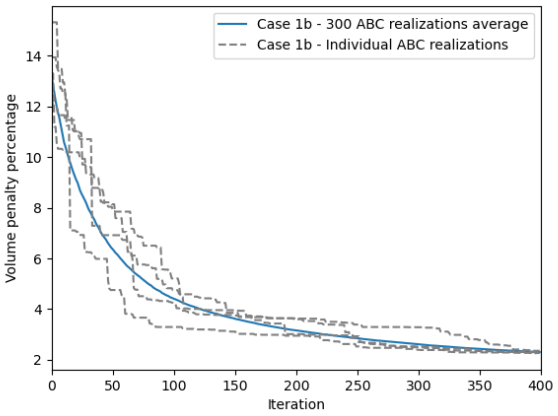


(c) MSPSO

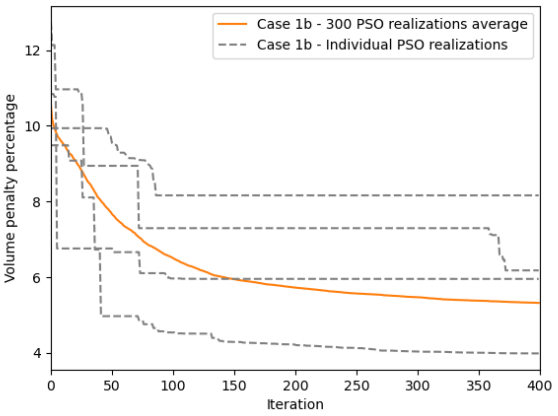


(d) 300 realization averages

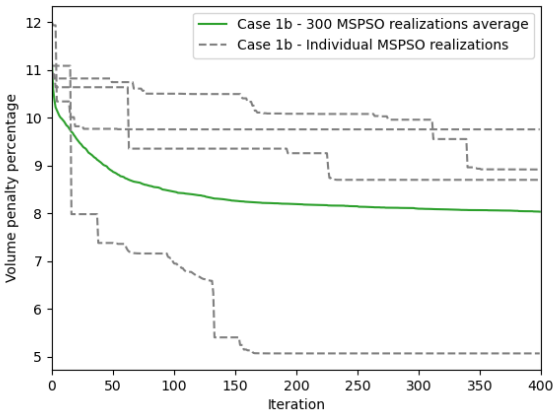
Figure B.1: Individual realizations and 300 realization averages of convergence of ABC, PSO and MSPSO hybrid method variants for case study 1a



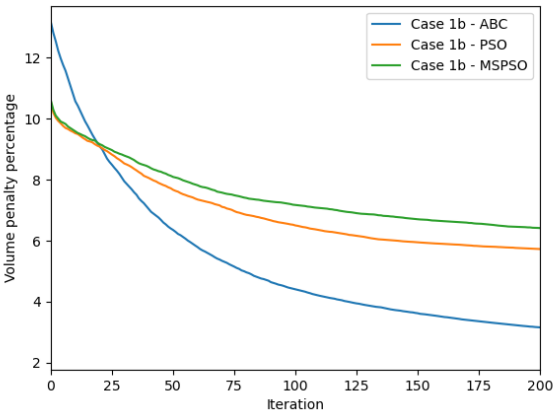
(a) ABC



(b) PSO

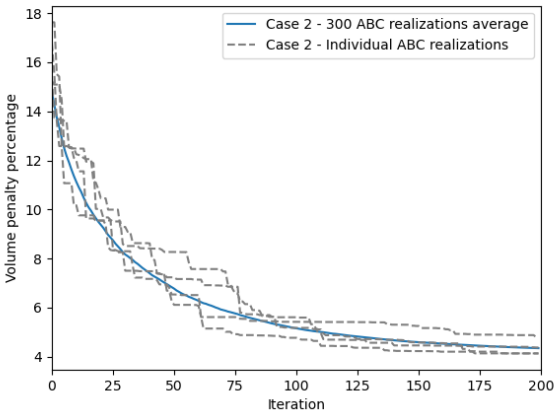


(c) MSPSO

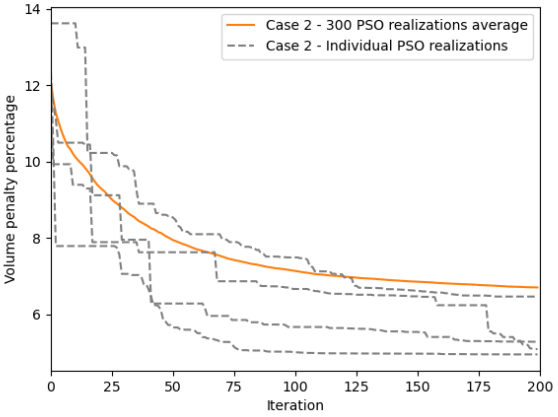


(d) 300 realization averages

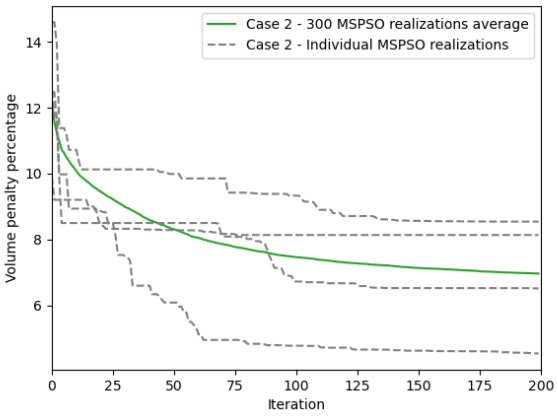
Figure B.2: Individual realizations and 300 realization averages of convergence of ABC, PSO and MSPSO hybrid method variants for case study 1b



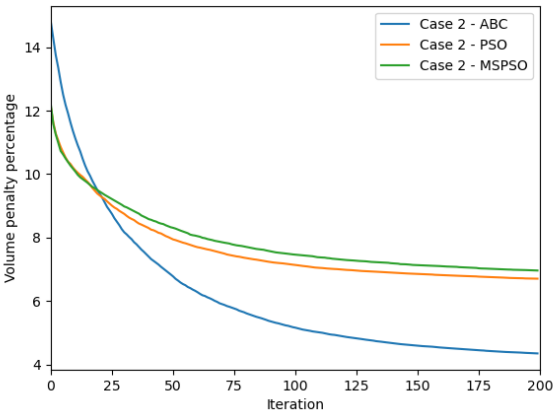
(a) ABC



(b) PSO



(c) MSPSO



(d) 300 realization averages

Figure B.3: Individual realizations and 300 realization averages of convergence of ABC, PSO and MSPSO hybrid method variants for case study 2