Diploma Thesis

Metamodel assisted optimisation of glued laminated timber systems by reordering laminations using metaheuristic algorithms

submitted in satisfaction of the requirements for the degree of
Diplom-Ingenieur
of the TU Wien, Faculty of Civil Engineering

Diplomarbeit

Metamodel unterstützte Optimierung von Brettschichtholz durch Umordnen der Brettlagen unter Verwendung von metaheuristischen Algorithmen

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines
Diplom-Ingenieurs
eingereicht an der Technischen Universität Wien, Fakultät für Bauingenieurwesen

von

Sebastian Pech, BSc

Matr.Nr.: 01126795

unter der Anleitung von

Univ. Ass. Dipl.-Ing. Dr.techn. **Georg Kandler**
Ass. Prof. Dipl.-Ing Dr.techn **Josef Füssl**
Univ. Prof. Dipl.-Ing. Dr.techn. DDr.h.c. **Josef Eberhardsteiner**

Institut für Mechanik der Werkstoffe und Strukturen
Technische Universität Wien
Karlsplatz 13/202, A-1040 Wien

Wien, im September 2017 _____

Danksagung

Ich möchte mich an dieser Stelle bei allen Personen bedanken die mich bei der Umsetzung dieser Diplomarbeit und während meines Studiums unterstützt haben. Besonders möchte ich meinem Betreuer Dipl.-Ing. Dr.techn. Georg Kandler danken, der mir das Diplomarbeitsthema vorgeschlagen hat und mich während der Umsetzung über alle Erwartungen hinaus unterstützt hat. Die Zusammenarbeit mit ihm hat mir immer Freude bereitet und ich konnte im großen Maße davon profitiert.

Ass. Prof. Dipl.-Ing. Dr.techn. Josef Füssl möchte ich für die Unterstützung in den letzten Wochen der Diplomarbeit und dem Zurverfügungstellen eines Arbeitsplatzes für die Zeit der Umsetzung danken. Ihm und Univ.Prof. Dipl.-Ing. Dr.techn. DDr.h.c. Josef Eberhardsteiner danke ich für die Chance am IMWS weiterhin Forschung zu betreiben.

Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Günther Raidl möchte ich für die Beratung und Tipps zum Thema metaheuristische Algorithmen danken die besonders in der Anfangsphase sehr hilfreich waren.

Bei Pivatdoz. Dipl.-Ing. Dr.techn. Christian Schranz MSc bedanke ich mich, dass ich während meines Studiums als Studienassistent am EDV-Zentrum tätig sein durfte und ich im Zuge dessen ihn und andere Kollegen am EDV-Zentrum als Freunde gewinnen konnte.

Auch möchte ich mich bei meinen Eltern Sabine und Anton bedanken ohne die es mir nicht möglich gewesen wäre dieses Studium zu machen. Meiner Freundin Lea gebührt besonderer Dank da sie die Person war mit der ich während meines Studiums die meiste Zeit verbracht habe und die mich in allen Phasen ertragen und unterstützt hat.

Kurzfassung

Ein übliches Verfahren zur Optimierung des Tragverhaltens und des Rohstoffeinsatzes bei Brettschichtholz (BSH) ist die Konstruktion von kombinierten BSH Trägern. Bei diesem Verfahren werden stärkere Bretter in den äußeren und schwächere Bretter in den inneren weniger belasteten Trägerlagen verbaut. Grundsätzlich ist diese Herangehensweise sinnvoll, allerdings biete sie Raum für Verbesserungen. Besonders die Vernachlässigung der Holzmorphologie eines Brettes (z.B. Äste und deren Lage) und die endgültige Position in einem BSH Träger sind von großer Bedeutung da eine korrekte Beurteilung von Schwächungen im Holz nur in Kombination mit der tatsächlichen Belastungssituation möglich ist. Beispielsweise kann diese Vernachlässigung dazu führen, dass ein Ast die Festigkeitsklasse des gesamten Brettes reduziert, obwohl er im BSH Träger in einem gering beanspruchten Bereich liegt und daher keinen Einfluss auf das Tragverhalten des Gesamtsystems des Trägers hat.

Aus diesem Grund war das Ziel dieser Arbeit die Entwicklung einer Methode zur Optimierung von BSH, die sowohl die mechanischen Eigenschaften als auch die tatsächlich auftretenden Spannungen berücksichtigt. Dafür werden die Träger mit einem zweidimensionalen Finite Elemente (FE) Modell analysiert, welches die Ermittlung der vorhandenen Spannungs- und Verzerrungsfelder erlaubt.

Da der Berechnungsaufwand bei dieser Optimierungsaufgabe mit der Anzahl an Trägern und Brettern schnell wächst, ist es erforderlich spezielle Algorithmen aus dem Bereich der Metaheuristik zu verwenden. In dieser Arbeit finden Local Search, Iterated Local Search, Tabu Search und genetische Algorithmen Verwendung. Anfangs werden die Algorithmen anhand eines vereinfachten Problems unter der Annahme homogener Materialeigenschaften bewertet. Im nächsten Schritt wird auf Basis dieses Modells die Lösbarkeit des Problems mittels deterministischer Algorithmen, anstelle der nicht-deterministischen metaheuristischen Methoden diskutiert.

Um die eigentliche Problemstellung (mit inhomogener Steifigkeitsverteilung) innerhalb eines vertretbaren Zeitraums lösbar zu machen wird die aufwändige Berechnung des FE Modells durch Verwendung eines Metamodells umgangen. Die Ergebnisse der Optimierungsdurchgänge werden statistisch ausgewertet, was die folgende Auswahl von geeigneten Lösungsmethoden zulässt: Iterated Local Search eignet sich zum Finden von einigermaßen guten Resultaten in kurzer Zeit. Genetische Algorithmen eignen sich zum Finden sehr guter Resultate, benötigen dafür aber mehr Rechenzeit.

Im Vergleich zu den üblichen Methoden der Herstellung von BSH ist es möglich mit den genannten Algorithmen bei gleicher Belastung Träger mit durchschnittlich 15 bis 20 % geringerer Durchbiegung zu konstruieren. Das bedeutet, dass unter Verwendung der selben Bretter die maximale Durchbiegung des schlechtesten Trägers um diesen Wert geringer ist. Zusammenfassend lässt sich sagen, dass die genannten Optimierungsverfahren auf die Problemstellung anwendbar sind und gute Ergebnisse in einem vertretbaren Zeitraum liefern. Darüber hinaus ist es aufgrund der allgemeinen Formulierung der Algorithmen möglich sie im Bereich des Ingenieurholzbaus auf ein breites Feld von Optimierungsaufgaben anzuwenden.

Abstract

A common approach for optimising the load-bearing behaviour of glued laminated timber (GLT) beams, with respect to an efficient use of the raw material, is producing combined GLT beams. Thereby, stronger boards, categorised based on a preceding strength grading method, are used for the outer layers of the beam, whereas weaker boards are used to fill the less stressed inner layers. This method, however, leaves room for improvement. Especially the omission of the real morphology of a board (e.g. knot groups and their position) and their location in the final beam setup is significant, since only this information together with the actual loading situation allows for a proper evaluation of weaknesses in the GLT beam. For example, a certain knot which reduces the strength grading class of a single board might be located in a not highly stressed region in the GLT beam and, thus, is actually negligible when considering its load-bearing behaviour.

For this reason, the objective of this thesis had been to develop an optimisation strategy for GLT beams, able to take actual mechanical property distributions as well as the occurring stress states in the final GLT beam within each individual board into account. To achieve this, the GLT beams are analysed using a two dimensional finite element (FE) model, giving access to the strain and stress field of each wooden board.

Subsequently, this information is exploited to find optimal GLT beam setups out of a defined sample of wooden boards. As the complexity and the computational effort of this combinatorial optimisation task quickly increases with the number of beams and wooden boards, a class of special algorithms, namely metaheuristic search methods, where introduced. In particular, local search, iterated local search, tabu search, and genetic algorithms where considered and are discussed in detail. In a first step, the algorithms are assessed on a simplified problem, which assumes homogeneous material properties for each board and, thus, allows the usage of beam theory. Next, based on this simplified model, the solvability by deterministic algorithms, instead of metaheuristic, non-deterministic algorithms is discussed.

In order to solve the original problem (with inhomogeneous stiffness distributions) within a reasonable time, the evaluation of the computationally costly FE model is bypassed by defining two types of metamodels, which are capable of approximating the FE model's results after an initial training phase on previously calculated results. All algorithms are tested multiple times, allowing a statistical validation of each method. This validation results in a preference for iterated local search, as an algorithm being capable of quickly finding moderately good results, and genetic algorithms, being capable of finding good results, however needing more computation time.

Comparing the results obtained from various optimisation approaches to commonly used methods within the production of GLT beams, on average an improvement of 15 to 20 % could be obtained, meaning that by using the same sample of wooden boards, the maximum deflection of the worst GLT beam is smaller by this value. Summarised, it can be said that the used metaheuristic search methods are applicable to this optimisation task and deliver good results within a reasonable time. Furthermore, due to the general nature of the proposed algorithms and definitions, they are applicable and expandable to a wide range of different optimisation tasks in timber engineering.

Contents

Chapter 1

Introduction

1.1 Motivation

Throughout the previous years, wood consequently gained importance in fields of civil engineering where usually mainly steel and concrete constructions were used. Especially buildings like the *HoHo* in Vienna, a 24 storage wooden high-rise building, demonstrate the capabilities of wood as a construction material. Besides its remarkable qualities in this concern, wood is a naturally grown resource, thus it is not only contributing to lowering CO_2 emissions in comparison to steel- or concrete production, but rather it even reduces the CO_2 content of the air. Therefore, under the assumption of sustainable forestry, focusing on using wood as a building material can help tackling nowadays climate change.

However, wood exhibits a quite complex mechanical behaviour, which is difficult to constitute in a mechanical model. Therefore, existing design rules are often based on empirical findings, which, especially in terms of efficiency, are unsatisfactory. In the scope of this thesis particularly the process of visual strength grading is discussed. Visual strength grading is a practice, still commonly used, where a wooden board is categorised based on visual characteristic [26] like

- knot sizes, -locations, -groups,
- fibre angles,
- location of the pith,
- width of annual tree rings,
- cracks,
- width of rough edges,
- curvature of the board,
- oxidative discolouration, rot and
- insect damage.

The process of visual grading of timber boards is regulated in ÖNORM DIN 4074-1 [39]. Especially the first two characteristics, concerning knots and fibre angles, mainly define the grading class

a wooden board is assigned to. The actual strength class of a timber board is allocated to the grading class based on ÖNORM EN 1912 [41].

The task of optimising the load-bearing behaviour of GLT beams is commonly performed by producing combined GLT beams, regulated by ÖNORM EN 14080 [40]. Herein, based on the previously described classification, stronger boards are used for the outer layers of the GLT beam, whereas weaker boards are used to fill the less stressed inner layers. This strategy, however, leaves room for improvement, and particularly the following limitations have to be considered.

1. Currently, the classification methodology considers the spatial variability of mechanical properties within wooden boards to a limited degree. However, knots and the resulting fibre deviations lead to strong localised effects and spatial fluctuations of stiffness and strength.

2. By using beam theory and assuming a homogeneous distribution of mechanical properties along wooden boards for distinguishing areas of high and low stresses within the beam, the actual structural behaviour of the beam cannot be represented very accurately.

3. The strength class is determined without considering the actual resulting stresses on the lamella in the final structure. Thus, certain classification criteria might lead to a reduction of the strength class of a lamella, although their impact within the final beam is neglectable.

Therefore, the grading process and the design process of GLT beams should be combined in order to better utilise timber boards. Consequently, the aim of this work is to, subsequent to the visual grading process of a given amount of timber boards, provide a desired amount of GLT beam designs which exhibit, under the given conditions, an optimal load-bearing behaviour. The improvement of the load-bearing behaviour is achieved by reordering lamellas such that the deflection of every beam is as small as possible. Figure 1.1 depicts the process from unordered stacked wooden boards to optimally constructed GLT beams. In this thesis, mainly the various optimisation methods for finding solutions to the stated optimisation problem are assessed and implemented. Finally, variations in the problem definition are discussed and the actual improvement compared to commonly used methods for the construction of GLT beams is determined.

The method proposed in this thesis utilises the data acquired during the visual grading process to model the mechanical behaviour of each timber board. Thus, the mechanical properties of each board are determined individually and not defined through a given class, hence spatial stiffness fluctuations are considered. Moreover, a two-dimensional FE computation is performed to determine the strains and stresses of each individual wooden board within the GLT beam assembly. This method provides a basis for a more elaborate optimisation of the load-bearing behaviour of GLT beams.

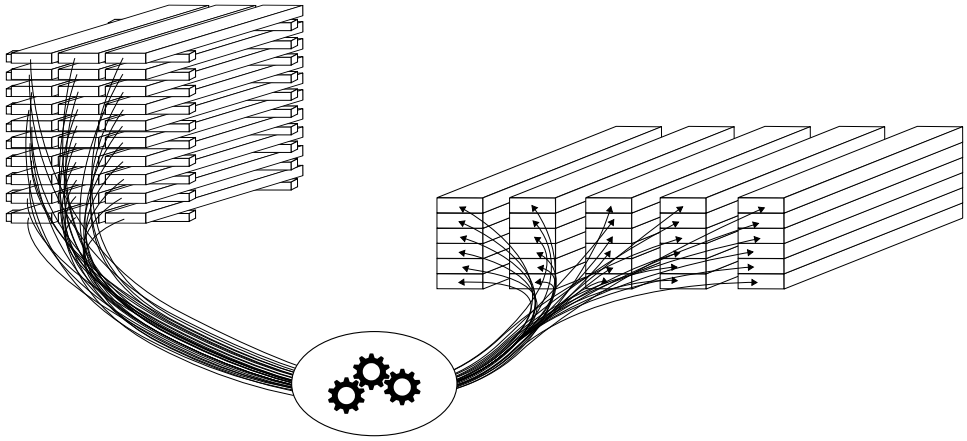


Fig. 1.1: Problem illustration from unordered boards to assembled beams. Finding an optimal algorithm to do this represents represents the general objective of this thesis.

1.2 Problem description

As indicated in Figure 1.1, the problems investigated throughout this thesis are of combinatorial nature. The goal is to improve the structural behaviour of a set of beams only by means of reordering the lamellas within the beams. In the following, a defined arrangement of lamellas forming multiple beams will be referred to as beam setup.

In this context, numerous different optimisation tasks for GLT beams can be formulated:

- Find an arrangement of n_l lamellas within n_b beams, where every beam consists of $n_{b,l}$ lamellas and $n_l > n_b \cdot n_{b,l}$, which minimises the maximum occurring deflection of a GLT beam within the beam setup.
- Find an arrangement of n_l lamellas within n_b beams, where every beam consists of $n_{b,l}$ lamellas and $n_l = n_b \cdot n_{b,l}$, which minimises the maximum occurring deflection of a GLT beam within the beam setup.
- Find an arrangement of n_l lamellas within one beam, which minimises the maximum deflection of the beam.

The following sections give an insight into the complexity of the stated problems. The basic concept and general considerations are explained by means of an example with $n_l = 50$, $n_b = 5$, and $n_{b,l} = 10$. This example is extended and explained in a more detailed manner in Section 5.1.

1.2.1 Investigating the problem

Considering the most general case for the problem described in Section 1.2, where $n_l \geq n_b \cdot n_{b,l}$, the number of possible combinations for picking $n_b \cdot n_{b,l}$ lamellas out of n_l lamellas, regarding their order is given by

$$\binom{n_l}{n_b \cdot n_{b,l}} \cdot n_l! = \frac{n_l!}{(n_l - n_b \cdot n_{b,l})!}. \tag{1.1}$$

Considering that individual boards show a inhomogeneous stiffness distribution in longitudinal direction (details in Chapter 2), the number of possible combinations must be reconsidered by the fact that one lamella can be build into a beam in two ways. This leads to another $2^{n_b \cdot n_{b,l}}$ possible combinations.

The resulting list of orders of lamellas can be reduced by considering that the order of the beams inside this list does not affect the result in any ways. Since the number of possible combinations of n_b beams is $n_b!$, the resulting equation describing the number of combinations is

$$\frac{2^{n_b \cdot n_{b,l}} \cdot n_l!}{n_b! \cdot (n_l - n_b \cdot n_{b,l})!}. \tag{1.2}$$

For the stated example, taking into account that $0! = 1$, the number of distinct combinations is $\approx 2.8536 \times 10^{77}$. Referring to the benchmark tests for the FE model described in Section 6.1, the average evaluation time of one beam is 292.3 ms. Therefore, the computation time for evaluating all combinations would be about 2.64×10^{69} years. This makes it practically impossible to determine the optimal result based on pure enumerative algorithms. Thus, optimisation techniques are needed which are able to deliver near optimal results in a reasonable time frame.

1.3 Structure of the thesis

Chapter 2 gives an insight into current state of the art material models for timber boards. Chapter 3 discusses various optimisation methods based on the problem description given in this chapter. The actual implementation of the used algorithms is described in Chapter 4. The application of those methods to a problem with reduced complexity is given in Chapter 5, whereas the actual application to the original problem is given in Chapter 6. Chapter 6 also shows the effects of different optimisation tasks and contains a discussion of the obtained results. Concluding remarks and an outlook on future improvement possibilities are given in Chapter 7.

Chapter 2

Material model for timber boards

For a sound mechanical model of GLT, the mechanical behaviour of the "basic components", the individual timber boards, has to be determined. Numerous approaches incorporate extensive experimental investigations ([13], [14]) to predict the mechanical properties of timber boards based on morphological parameters – so-called indicating properties – such as visible knot area. In recent years, however, numerical approaches became more and more popular for predicting the effective mechanical properties of timber boards [22, 23, 34]. These approaches require detailed knowledge of the knot morphology to identify the stiffness properties of individual timber boards.

The discussed data acquisition approach relies on laser scan data, which is obtained during the grading process of individual wooden boards. The underlying utilized effect is the so-called tracheid-effect, which describes the light propagation on a wooden surface. Due to the orthotropic fibre structure of wood, a concentrated light source, e.g. a laser, spreads differently parallel to the fibre than perpendicular. Thus, a laser dot deforms to an elliptical spot and reveals the major material axis.

Based on the measured fibre angles, two different approaches are pursued at IMWS to obtain the localised stiffness tensors:

- Directly using the wood fibre angles [28]. This approach is similar to Oscarsson et al. [42], Petersson [43] and Olsson et al. [38].
- Reconstructing the knot morphology to perform 3D linear elastic FE analysis [29].

The direct procedure presented in [28], where the wood fibre angles are directly used to obtain longitudinal stiffness distributions by transforming the clear wood stiffness tensor, additionally uses an empirical model to determine the dive-angle [28, 37, 48]. The stiffness tensor is obtained in each measurement point from a micromechanical model based on a multi-scale homogenisation procedure [24]. Homogenising the values over each cross section results in a so-called stiffness profile for each board, describing the variation of the local modulus of elasticity E in the longitudinal direction x, hence $E \equiv E(x)$. By using these stiffness profiles in combination with a linear 2D FE model, an accurate mechanical model for predicting the effective stiffness of GLT beams was developed by Kandler et al. [28].

For this work, stiffness profiles based on the latter approach are employed. Therefore, in the following a brief summary of the procedures is given.

From the above mentioned laser scan data knot areas are identified on all four surfaces of the board. Through an optimisation scheme the most probable knot arrangements are identified. A detailed explanation of this reconstruction algorithm is presented in [29]. The main aspects of the algorithm can be outlined as follows:

1. First, a three-dimensional fibre angle is computed through superposition of the in- and out-of-plane fibre angles . Then individual knot areas are determined by comparing this fibre angle in every point with a threshold value.

2. The pith location is estimated by fitting circles to the year rings, which are obtained from photographs of the cross section, at both ends of the board. The pith is reconstructed by connecting the arithmetic means of the centre points of those circles with a linear curve.

3. The actual knot is then reconstructed by a rotationally symmetric cone, defined by a cone apex, a knot vector, and an opening angle. In a first step, the knot axis is reconstructed by connecting all pairs of knot areas, obtained from step 1, which do not share the same face of the board. For each of these knot axis candidates the quality of the fit is calculated based on the normal distance between the axis and the pith (step 2) and the distance between the two knot areas. Then the most promising knot axis are picked through an iterative scheme, by removing knot axis candidates with a high quality fit from the pool of axis candidates. In this way, in every step the most likely knot axis and all other candidates related to one of the knot areas connected to the selected axis are removed from the pool of candidates.

   This scheme is repeated until all axis are assigned. In case there are still knot areas remaining in the pool, it is assumed that those knots only partly penetrate the board. Their knot axis are reconstructed by estimating the angle of already reconstructed knot axis.

4. The quality of this deterministic algorithm is further improved by minimising the reconstruction error (difference between the actual knot areas and the knot areas obtained in step 4) through a simulated-annealing optimisation scheme.

Based on these geometries, the 3D fibre angles within the volume of the board are computed. The procedure utilizes the so-called grain-flow analogy [16, 17], which is based on the theoretical behaviour of a laminar fluid flowing around elliptical obstacles. The analogy lies in the assumption that knots act as obstacles and fibres, in the radial-tangential plane, represent trajectories of the laminar fluid.

The resulting knot geometries and 3D fibre angles are passed to a linear elastic 3D finite element analysis, where each knot group is loaded under a tension to estimate its effective stiffness. The boundary conditions are applied such that one end of the knot group is fixed and the other end is exposed to a prescribed deformation. The remaining boundary conditions were defined such that the specimen could expand freely in the lateral directions. The effective modulus of elasticity is computed from the resultant forces. Subsequently, this procedure is applied to all knot groups, yielding a piecewise constant stiffness profile, as can be seen in Figure 2.1.

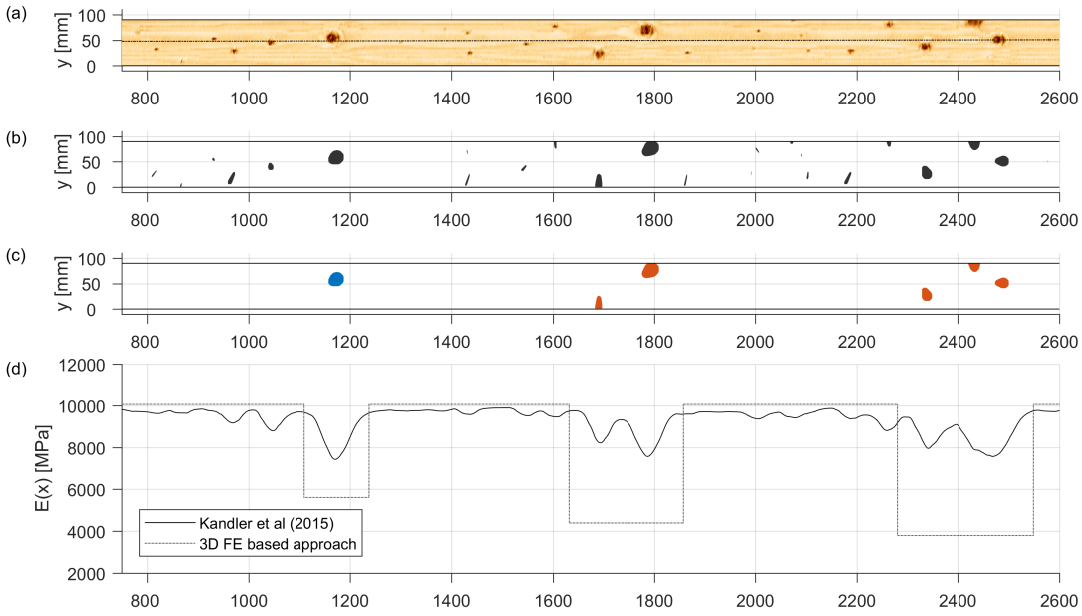


Fig. 2.1: **(a)** Top view of a board. **(b)** Knots including small knots. **(c)** Knots without small knots. **(d)** Resulting stiffness profiles for both described methods.

Chapter 3

General discussion of optimisation methods

3.1 Optimisation algorithms

Dependent on the type of problem a number of different optimisation strategies are available. Goldberg [21] reviews three general types of search and optimisation techniques:

- Calculus-based methods which, can be further subdivided into
  - indirect and
  - direct methods,
- enumerative methods, and
- random search algorithms, such as e.g.
  - simple random search,
  - genetic algorithms,
  - particle swarm optimisation,
  - tabu search,
  - simulated annealing,
  - iterated local search,
  - ...

Indirect calculus-based methods usually seek local extrema by solving the equations resulting from setting the gradient of the given function to zero [27]. Direct calculus-based methods evaluate the local gradient at a function value and start moving in a related direction[1][33].

The downside of those methods is, that in order to be applicable, the optimisation problem must be formulated in terms of a piecewise continuous differentiable function. As it is not possible to formulate the stated problem in such a manner, calculus-based methods are unsuitable.

Enumerative methods take a different, fairly simple approach by just evaluating every possible solution in the search space. On the one hand, this approach guarantees the identification of the global optimum, on the other hand, obtaining results in a reasonable time frame limits this

[1] These kind of algorithms are often referred to as "Hill-Climber" algorithms

approach to small search spaces or a low number of variables. For most optimisation problems this limitation is so severe, that enumerative methods are often infeasible (See also Section 1.2.1).

This leaves random search algorithms as only viable solution for solving the described problem, by evaluating the objective function at random points and saving the best resulting value. A subclass of random search algorithms are the so-called metaheuristic algorithms. Blum et al. [5] summarised the fundamental properties of metaheuristic approaches as followed:

- Metaheuristic algorithms (MAs) are used to guide a search process.
- The focus of MAs rather lies on exploring the search field to find a solution close to the global optimum than to focus the search on only improving a solution and maybe converging towards a local optimum.
- MAs are usually non-deterministic[2].
- MAs are not specifically bound to a certain class of problems.

When working with metaheuristic algorithms it is very important to preserve the balance between exploration and exploitation. Exploration corresponds to a rough evaluation of the search space, whereas exploitation is necessary to refine an area of good solution towards the optimum.

In Figure 3.1, grey circles visualise wide-spread solutions in different fields of the search space, thus representing exploration, whereas the arrows depict the exploitation capabilities needed to refine the found solutions. It can be seen that neither exploration- nor exploitation strategies alone deliver values close to the global minimum of the function. By combining both approaches it is possible to find such solutions within a sensible computation time. Algorithms with a low capability of exploring and thus a small insight on the search space, tend to converge against local optima, whereas algorithms which search a wide field of the solution space, resemble random walks and perform poorly on finding an optimum.

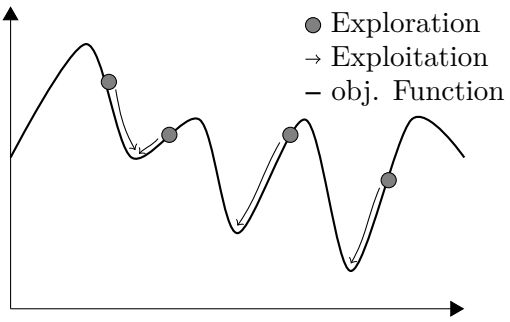


Fig. 3.1: Difference between the concepts of exploration end exploitation

Gendreau et al. [20] and Blum et al. [5] give a detailed insight into the implementation of various metaheuristic algorithms. As the stated problem is a combinatorial optimisation (CO)

[2]Floyd [15] introduces "non-deterministic" as a description for algorithms with a kind of "free will". Those algorithms are not random but can produce a different outcome on different runs with equal input.

problem, the used metaheuristic must be able to cope with solutions encoded with discrete variables. Especially for CO problems local search (LS), tabu search (TS), and genetic algorithms (GAs) can yield viable solutions.

3.2 Metaheuristic algorithms

In the following sections, the most relevant algorithms for the problem described in Section 1.2 are discussed briefly to provide the reader, without any experience in this field, with a basic knowledge to comprehend the assumptions and findings in the following sections. Furthermore, commonly used vocabulary is introduced.

Details on the implementation and adaption for the stated problem are given in Chapter 4 and Section 4.1.

3.2.1 Local search and iterated local search

The basic LS is the simplest algorithm among the discussed. Lourenço et al. [33] describes LS as follows: The target is to minimise the objective function f for a CO problem. Ω denotes the set of all possible solutions s for the given problem.

As the name LS implies, the search procedure is not performed globally for all $s \in \Omega$, rather a local search in a neighbourhood $\Gamma \subset \Omega$ around s is performed.

Each step of the LS results in a new s^* and $\forall s : f(s) \geq f(s^*)$ where $s^* \in \Gamma$. The local optimum can be found by recurring calls to the LS algorithm until a termination criterion is met.

LS procedures are inherently of deterministic nature, as f is evaluated for all s^* in Γ. As this procedure can be computationally costly, LS is usually endowed with a heuristic component in form of only evaluating a random subset of Γ.

As Blum et al. [5] and Lourenço et al. [33] point out, the problem with LS algorithms is that they strongly depend on how s, f and Γ are defined and that they can easily be trapped in a local minimum.

Nonetheless LS performs well on the exploitation of areas of good solutions, therefore LS is often combined with another MA which is capable of escaping local minima.

One implementation is the so-called iterated local search (ILS) mentioned by Lourenço et al. [33]. The basic idea is that multiple LS algorithms search on different neighborhoods Γ_i. When a local minimum occurs a perturbation is applied which results in a new neighborhood Γ_i'. The search is then continued within Γ_i' until $\forall s' : f(s') \geq f(s'^*)$ for $s' \in \Gamma_i'$ is satisfied. The perturbation is accepted if s'^* passes an acceptance test.

In the context of the current problem, a neighbourhood Γ_i could be defined based on a single beam. Thus, the initial LS algorithms are limited to optimising one beam only. The perturbation can then be applied in form of performing an exchange of lamellas between two beams. Therefore, a new neighbourhood Γ_i' is generated, which allows the LS to reach alternative areas of the solution space.

3.2.2 Tabu search

As described by Bianchi et al. [3], TS is in simple terms an improved type of LS. The basic idea is to allow LS to overcome local optima by allowing non-improving moves and simultaneously preventing cycling back to previous solutions [19]. The basis on which TS operates are tabus and aspiration criteria.

Tabus are moves or changes affecting the current solution which are prohibited for a certain number of steps also referred to as the tabutenure [19]. The purpose of a tabu is to, as already mentioned, prevent the algorithm from backtracking previous solutions.

Considering such moves in the context of the current problem, tabus can be formulated in form of

- beam setups or
- beams

or alternatively in form operations like

- swapping two distinct lamellas l_i and l_j or
- moving a lamella l_i back to a beam b_i.

The tabutenure is used to discard tabus after a defined period of steps to allow the algorithm to perform previously forbidden operations in a later state.

Aspiration criteria can be seen as a safety measure to allow the algorithm to revoke tabus to reach solutions which, for example are better than the previously known best solution.

Beside the newly introduced concepts TS works based on the best improvement [3] method, which distinguishes the algorithm from the concept LS is based on. LS accepts a solution s^* only if it matches the criterion $f(s^*) \leq f(s)$ where s denotes the best known solution, whereas TS accepts the best solution within the neighborhood even if it does not match $f(s^*) \leq f(s)$ as long as s^* does not violate any tabus.

3.2.3 Genetic algorithms

The main idea of GAs is borrowed from evolutionary theory. That is, given enough time, a population will adapt and improve over successive generations through concepts of natural selection, mutation and recombination. Thus, GAs are a subclass of evolutionary algorithms (EAs).

Goldberg [21] describes the fundamentals of GA as follows: The basis for every simple GA is a data structure called a chromosome. Putting the term chromosome into a more mathematical context, it could be represented by a vector or a list. The chromosome itself is constructed by genes which can take different values called alleles. The position at which a gene is situated within the chromosome is referred to as the gene locus.

The starting point for every GA is a population of individuals, usually represented by chromosomes, forming the first generation of solutions.

The operators involved in transition from one generation to the next are:

- selection,
- crossover and
- mutation.

During the selection phase, strong individuals, from the current population, are selected to form the next generation. The crossover phase generates the offspring based on the parent population and the mutation phase assures that the genetic diversity is maintained. The operators will be discussed in detail in Section 4.8.

Reeves [44] mentions additional concepts like elitism, where a portion of the best individuals is not replaced, and steady-state algorithms, where only one individual per generation is replaced.

Chapter 4

Implementation of methods for optimisation

4.1 General formulation of the stated problem in terms of combinatorial optimisation

To be able to apply the discussed algorithms to the stated problem, a general and abstract definition in a mathematical sense is needed. Based on the problem description given in Section 1.2, this definition can, independently of the optimisation algorithms, be formulated as follows: Let $\pi : L \to L$ be a permutation of the set of lamellas L and $g : l \mapsto l^*$ be a function which defines the orientation of a lamella l. Then $\pi^* = g \circ \pi$ returns a permutation of L with explicitly defined orientations. Let Φ be a map $\Phi : \pi^* \to B$ which maps the lamella permutation from lamella space into beam space B. Find a π^* which minimises the objective function $f(\pi^*)$.

This definition generalizes the problem description in Section 1.2. In most cases throughout the work f performs operations based on the function $q_{\max}$, which calculates the maximum deflection of a given beam b_i. Therefore, f is specialized for $q_{\max}$ as shown in Section 4.1.1.

The formulation for the map Φ in Section 4.1.2 is still kept general in terms of the problem description, as π^* is not further specialized. Φ is adapted for the usage of MAs.

4.1.1 Definition of the function $f(\pi^*)$

As stated above, f is considered to perform operations based on a function $q_{\max}$. The objective of this CO problem is to generate a beam setup where the value of $q_{\max}$ (maximum deflection of a beam) is as small as possible. This leads to the formulation

$$f(\pi^*) = \max \{q_{\max}(b_i) | b_i \in \Phi(\pi^*)\} . \tag{4.1}$$

An alternative formulation in terms of reducing the mean value of $q_{\max}$ for the beam setup would be possible. However, for practical reasons it is not applicable to the stated problem, as solutions would be possible where superior beams compensate inferior ones. This compensation is not granted within an actual construction. This statement is further exemplified in Section 4.2.1.

4.1.2 Definition of the map Φ

As will be discussed in subsequent sections, a vector or list is suitable for representing a lamella order for a beam setup. The map Φ defines how this vector is linked to the actual GLT beams.

As the input chromosome or input vector[1] is defined by the permutation π, the location within a beam can be defined by the gene locus or the index, as shown in Figure 4.1.

In the following, two different index-to-beam links will be discussed. From the described MAs from Section 3.2, LS, ILS and TS are independent of the linking type chosen for Φ. GAs on the other hand are considered to be sensitive to the definition of Φ due to the processes involved during crossover.

This is expressed in the so-called building block hypothesis described by Holland et al. [25] and Goldberg [21]. The building block hypothesis states that a chromosomes fitness value is vastly influenced by short subsequences of the chromosome. A detailed explanation and reasoning for the described variations of Φ is given in Section 4.1.2.1.

Let L_i^* denote the vector of lamellas for beam b_i. Regarding the building block hypothesis L_i^* can be defined as

$$L_i^* = \left\{ \pi^*(k) | k \in \{ i + j \cdot n_b \}_{j=0}^{n_{b,l}-1} \right\}, \tag{4.2}$$

or alternatively without considering building blocks as

$$L_i^* = \left\{ \pi^*(k) | k \in \{ n_{b,l} \cdot (i-1) + j \}_{j=1}^{n_{b,l}} \right\}. \tag{4.3}$$

Subsequently the mapping based on Equation (4.2) will be referred to as chromosome type A and the mapping based on Equation (4.3) as chromosome type B. LS, ILS and TS will solely be based on Equation (4.3).

In the context of this formulation L_i^* equals b_i, as the necessary structural information for calculating the deflection is stored in $q_{\max}$. Therefore, regardless of which of the two definitions of L_i^*

$$\Phi(\pi^*) = \{ L_i^* |\ i \in \{1, \ldots, n_b\} \}. \tag{4.4}$$

Figure 4.1 visualizes the map from lamella space to beam space for both chromosome types for $n_l = 12$, $n_b = 3$ and $n_{b,l} = 4$ (In favor of the readability only some of the mappings are visualised).

4.1.2.1 Building block hypothesis

In the previous section one definition of Φ was made based on the so-called building block hypothesis [21, 25].

The building block hypothesis states that a chromosomes fitness value is vastly influenced by schemata with a short defining length and a higher than average fitness called building blocks.

In terms of GAs a schema [25] defines a basis for describing similarities of chromosomes.

As an example adapted to CO the schema `1, 2 ,* ,*` is a template for the chromosomes `1, 2, 3, 4` and `1, 2, 4, 3` . The defining length of a schema is the distance between the outer fixed genes. For the present example the defined length is 1.

[1] The terms input vector and input chromosome are in the current context synonymous. The term is switched in favor of the terminology of the currently discussed optimisation procedure.

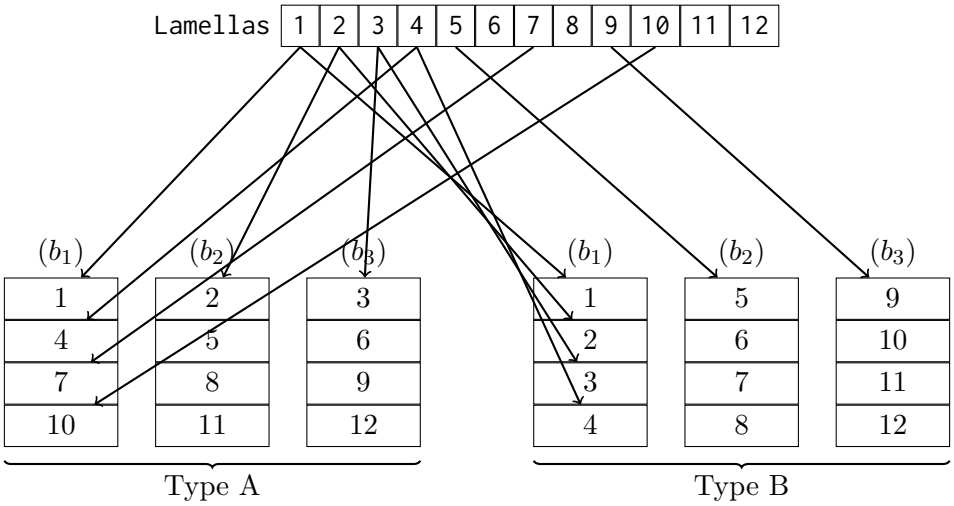


Fig. 4.1: Example mapping from a lamella vector or chromosome to beams using chromosome types A and B

The advantage of such short schemata over widespread ones is that their chance to survive a crossover is higher due to the smaller amount of crossover points which separate the schema.

Figure 4.2 shows two different chromosomes A and B. The asterisks are used as place holders for arbitrary alleles, which in the context of the building block hypothesis are not of concern. The two genes with values 1 and 2 are in this example genes which, when combined contribute an above average share on the chromosomes fitness. By performing a crossover operation in most cases the chromosome is cut at a random position and recombined with another one. In cases where the cut is made between the important genes they end up in two different new chromosomes which means a probably good solution was destroyed. All possible locations for cutting chromosomes A and B which destroy the schema are visualised by arrows. Clearly, chromosome B has more destructive cut possibilities than chromosome A. Thus, the chance that the shorter schema survives the crossover is higher than the chance that the longer schema survives.

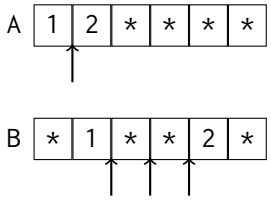


Fig. 4.2: Building block hypothesis: Chromosomes with schemata of higher than average share on the chromosomes fitness with different defining lengths.

It is arguable, that by the same reasoning coincidentally grouped “bad” lamellas are saved from being destroyed by crossover. This is certainly true, however, such schemata are likely to be removed during the selection phase due to under average fitness values.

The concept of building blocks can be transferred to GLT beams as follows: Supported by the experiments made by Fink et al. [14] and Serrano et al. [47], and as shown in Section 4.2, the further away a lamella is located from the beams centre of mass, the higher the impact on the load-bearing behaviour. Therefore, for minimising Equation (4.1) it is important that "good lamellas"[2] end up located furthest from the beams centre of mass.

In terms of the build block hypothesis the conclusion can be drawn that in order to save good solutions for Equation (4.1) from being destroyed by crossover the important locations within a beam should be grouped together. Thus, resulting in the definition of Equation (4.2).

4.2 Simplification of the original problem

In order to gain a general insight on MAs and validate them as a possible optimisation procedure, a problem with reduced complexity is investigated.

A possible approach towards reducing the complexity is neglecting the variability of the longitudinal stiffness, represented by the so-called stiffness profile $E(x)$, mentioned by Kandler et al. [28], and instead considering each lamella with a constant stiffness $E(x) = E$.

As this optimisation problem solely serves the purpose of verifying MAs, the bending stiffness of the overall beam setup is maximised, which in this simplified case is equal to minimising the maximum displacement.

For the special case $E_i(x) = E_i$, the orientation of the lamellas becomes irrelevant, hence $\pi^* \equiv \pi$. Therefore, the bending stiffness of one beam is defined as

$$EI_b(b_i) = \sum_{l \in L_i \equiv b_i} \left(E_l \cdot I_l + E_l \cdot A_l \cdot z_{l,i}^2 \right), \tag{4.5}$$

where A_l denotes the area of the cross section, I_l the second moment of area of the cross section and $z_{l,i}$ the distance of lamella l to the centre of mass of beam b_i. In the present example, both A_l and I_l are not only constant within each lamella, but equal for all lamellas, i.e. $A_l \equiv A$ and $I_l \equiv I$.

While the dimensions, cross sectional area A and stiffness values E_l are given, $z_{l,i}$ can be varied by reordering the lamellas.

The objective function for the entire beam setup, in analogy to Equation (4.1), is defined as

$$f(\pi) = \min \left\{ EI_b(b_i) | b_i \in \Phi(\pi) \right\}. \tag{4.6}$$

As can be seen from Equation (4.5), for maximising Equation (4.6) by reordering the lamellas, the term $E_l \cdot A_l \cdot z_{l,i}^2$ needs to me maximised. In other words, the higher E_l the higher $z_{l,i}$ should be.

This corresponds to the assumption made in Section 4.1.2.1 that "good lamellas" should be placed furthest from the centre of mass of the beam. In this context, Equation (4.5) allows a

[2] The term "good lamella" is kept intentionally vague because for the current context and formulation it is not important what is considered to be a "good lamella".

definition of the term "good lamella": A Lamella l_1 is assumed to be "better" than another lamella l_2 if $E_1 > E_2$.

4.2.1 Discussion of solution patterns

The following section tries to answer the questions if there is an algorithm which is capable of constructing a solution pattern for Equation (4.6) in a deterministic manner, thus rendering the need of using MAs useless.

Hypothesis 1 *The desired arrangement which maximises Equation* (4.6) *can be generated by sorting the lamellas in an descending order by their stiffness and placing them in an inward spiral over all beams.*

Considering the alternative objective of optimising the mean value of all beams' bending stiffnesses defined by Equation (4.5), the objective function can be defined as

$$f_A(\pi) = \frac{\sum_{b_i \in \Phi(\pi)} EI_b(b_i)}{n_b} \mathrel{\widehat{=}} \sum_{b_i \in \Phi(\pi)} EI_b(b_i). \tag{4.7}$$

Equation (4.7) can be transformed by substituting $EI_b(b_i)$ with Equation (4.5) to

$$f_A(\pi) = \sum_{b_i \in \Phi(\pi)} \sum_{l \in L_i \equiv b_i} \left(E_l \cdot I_l + E_l \cdot A_l \cdot z_{l,i}^2 \right). \tag{4.8}$$

$E_l \cdot I_l$ and A_l are independent of b_i, i.e. they will assume the same values regardless of how the lamellas are arranged in the beam setup. Thus, they can be factorized, which yields

$$f_A(\pi) = \underbrace{\sum_{l \in L} E_l \cdot I_l}_{\text{constant by means of } \pi} + \underbrace{A_l}_{\text{constant}} \cdot \sum_{b_i \in \Phi(\pi)} \sum_{l \in L_i \equiv b_i} \left(E_l \cdot z_{l,i}^2 \right). \tag{4.9}$$

The constant terms from Equation (4.9) can be neglected for the optimisation problem, which leaves the term

$$\sum_{b_i \in \Phi(\pi)} \sum_{l \in L_i \equiv b_i} \left(E_l \cdot z_{l,i}^2 \right) \tag{4.10}$$

to be maximised. The desired permutation π which satisfies the demand is clearly achieved by ordering the lamellas in such a way that the stiffness increases with the distance to the centre of mass of the beam.

Figure 4.3 exemplary shows the optimal arrangement of 15 lamellas spread over 3 beams. The lamellas are numbered according to their stiffness. Lamella 1 has the highest stiffness, lamella 15 the lowest.

This solution pattern is capable of finding the best possible solution for Equation (4.7), but fails on Equation (4.6) for unevenly distributed lamella stiffnesses.

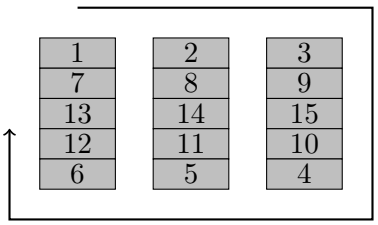


Fig. 4.3: Optimal arrangement of lamellas with constant longitudinal stiffness, based on Hypothesis 1

Assuming the following stiffness values $\boxed{\mathsf{E}_1,\mathsf{E}_2,\mathsf{E}_3,\mathsf{E}_4,\mathsf{E}_4,\mathsf{E}_4,\mathsf{E}_5,\mathsf{E}_6,\mathsf{E}_7}$ for 9 lamellas which shall be distributed over 3 beams $\boxed{\mathsf{b}_1,\mathsf{b}_2,\mathsf{b}_3}$ where $E_i > E_{(i+1)}$. The algorithm assigns the lamellas based on their stiffnesses as follows:

1. Top outer position: $E_1 \to b_1$, $E_2 \to b_2$, $E_3 \to b_3$
2. Bottom outer position: $E_4 \to b_3$, $E_4 \to b_2$, $E_4 \to b_1$
3. Middle position: $E_5 \to b_1$, $E_6 \to b_2$, $E_7 \to b_3$

After step 1 the beams can be sorted by their bending stiffness as follows: $b_1 > b_2 > b_3$.

Step 2 should regulate the stiffness descent from b_1 to b_3 but in this case, as all lamellas have equal stiffnesses E_4, the beam order does not change.

Step 3 should regulate the stiffness descent from step 2, but as there was no stiffness descent, once again the order $b_1 > b_2 > b_3$ is favoured, although the right solution to Equation (4.6) would have been to assign the lamellas the other way around, to compensate step 2.

Therefore, the conclusion can be drawn that the solution patter provided by Hypothesis 1 cannot be used to find a suitable solution for maximising Equation (4.6).

Hypothesis 2 *As lamellas with high stiffness values should be placed furthest from the centre of mass, the optimum location for each lamella can be defined only based on the lamellas stiffness.*

This hypothesis is based on the assumptions made in Hypothesis 1 but the algorithm is capable to overcome the disadvantages of prescribing the order in which the lamellas are distributed over the beams.

During the construction of the solution pattern the algorithm evaluates Equation (4.5) for the placed lamellas for each beam to decide which needs the current lamella the most.

To assure that every lamella is used at its most efficient position they are placed in an descending order by their stiffness value, starting from the outer layers of the beams. This restriction also follows Hypothesis 2, as the most efficient way of placing the lamellas (without further knowledge of the beam they will be located in) is the proposed order.

Assuming the following stiffness values $\boxed{2 \cdot \mathsf{E}_1,\mathsf{E}_1,\mathsf{E}_1,\mathsf{E}_1,\mathsf{E}_2,\mathsf{E}_2}$ for 6 lamellas which shall be distributed over 2 beams $\boxed{\mathsf{b}_1,\mathsf{b}_2}$ where $E_i > E_{(i+1)}$, the algorithm assigns the lamellas based on their stiffnesses as follows:

1. Top outer position: $2 \cdot E_1 \to b_1$, $E_1 \to b_2$

2. Bottom outer position: $E_1 \rightarrow b_2$, $E_1 \rightarrow b_1$

3. Middle position: $E_2 \rightarrow b_2$, $E_2 \rightarrow b_1$

After step 1 the beams can be sorted by their bending stiffness as follows: $b_1 > b_2$. Based on the order after the first step b_2 gets E_1. As the bending stiffness of b_2 now equals the bending stiffness of b_1 and b_1 offers a more efficient position, E_1 is assigned to b_1. This results in the order $b_1 > b_2$. The last step is based on the same reasoning as steps 2, therefore the final order is $b_1 > b_2$.

Let the distance of the outer lamellas to the centre of mass of the beam be 1. By evaluating Equation (4.5) for $I_l = 1$ and $A_l = 1$, the beam bending stiffnesses can be expressed by

$$EI_{b,1} = \underbrace{2 \cdot E_1 + 2 \cdot E_1}_{l_1} + \underbrace{E_1 + E_1}_{l_4} + \underbrace{E_2}_{l_6} = 6 \cdot E_1 + E_2 \quad \text{and} \tag{4.11}$$

$$EI_{b,2} = \underbrace{E_1 + E_1}_{l_2} + \underbrace{E_1 + E_1}_{l_3} + \underbrace{E_2}_{l_5} = 4 \cdot E_1 + E_2. \tag{4.12}$$

However, the solution which fulfills Equation (4.6) is obtained by assigning all lamellas with a stiffness value of E_1 to beam b_2 and the remaining ones to b_1, which leads to

$$EI_{b,1} = \underbrace{2 \cdot E_1 + 2 \cdot E_1}_{l_1} + \underbrace{E_2 + E_2}_{l_5} + \underbrace{E_2}_{l_6} = 4 \cdot E_1 + 3 \cdot E_2 \quad \text{and} \tag{4.13}$$

$$EI_{b,2} = \underbrace{E_1 + E_1}_{l_2} + \underbrace{E_1 + E_1}_{l_3} + \underbrace{E_1}_{l_4} = 5 \cdot E_1. \tag{4.14}$$

Both Equation (4.13) and Equation (4.14) give results that are greater than the one given by Equation (4.12). This confirms that the solution provided by Hypothesis 2 cannot be used to find a solution which reliably maximises Equation (4.6).

This observations further leads to the following conclusion: In order find an optimal solution to Equation (4.6), the algorithm must be able to place lamellas at locations where they are not at their full potential, which means it is not possible to construct a solution pattern solely based on lamella stiffnesses.

Therefore, even this simplified problem cannot be solved by using a purely deterministic algorithm. Hence it is inevitable to perform the optimisation based on MAs.

4.3 Vectorized representation of beams and lamellas

For solving the described problem numerous different algorithms for

- solving the actual optimisation problem or
- calculating the deflection of a beam or
- approximating deflections of beams

are needed. Thus, a universal representation is required, satisfying the demand of all algorithms.

In Section 4.1, possible solutions to the CO problem were assumed to be in form of permutations of a vector of lamellas. In Section 4.1.2, the map Φ was introduced which maps a vector of lamellas to a vector of beams, thus, effectively returning a vector of vectors of lamellas. Furthermore, the function g was introduced in Section 4.1 which defines the orientation of a lamella.

Taking all that into account a representation is needed which is capable of storing the lamella and its assigned orientation.

Throughout this work two different representations are used:

- a pure integer representation, capable of storing the bare minimum of data and
- a lamella data type, storing the entire data needed for one lamella.

In principle storing the entire data in one type is preferred over storing just its number. The advantage of this representation is, that in order to allow communication between the described algorithms no interface is needed, as all algorithms utilize the same lamella data type.

However, some of the used algorithms are not capable of dealing with a non-integer representation. Furthermore, as there is need for storing beams i.e. vectors of lamellas in hash tables [8, p. 219] (see Section 6.1) a more simplified and hashable representation is needed.

As already stated, the minimum amount of information needed to specify a lamella's usage are its id and orientation. For the described problem, the orientation can be one of two directions i.e. the lamella is flipped or not. It is possible to use the following integer representation for lamella i:

$$\text{Lamella } i : \begin{cases} 2 \cdot i, & \text{if flipped} \\ 2 \cdot i - 1, & \text{otherwise} \end{cases} \tag{4.15}$$

Essentially this means every lamella can be represented by the pair of the i-th even and odd number.

4.4 Implementation of local search

As described in Section 3.2.1, the algorithm for performing a LS is of rather simple nature, as is it just needs to pick solutions s^* from a neighbourhood Γ and validate them against the best known solution s. The newly obtained solution is then only accepted if it matches the criterion $f(s) \geq f(s^*)$.

In terms of the simplified example from Section 4.2, an optimal solution is found by maximising Equation (4.6). However, since LS searches for a minimum, f is defined as the inverse of Equation (4.6) (In this case using the inverse of Equation (4.6) is applicable since $EI_b(b_i) > 0$).

The space for feasible solutions s and s^* is L. The neighbourhood Γ around the current solution s is generated by performing every possible swap of two lamellas within s, thus generating beam setups with a high resemblance to s. The number of possible swaps for n_l lamellas which generate a new beam setup, and thus the size of the neighbourhood Γ, is obviously $\frac{(n_l-1)\cdot n_l}{2}$.

As the algorithm should be tested for the actual optimisation problem, it is already adapted to a computationally intensive objective function, by adding a stochastic component as follows: To reduce the number of calls to the objective function f, LS uses the first randomly picked solution s^* for which $f(s) \geq f(s^*)$. Of course the downside of this approach is that probably good solution are overlooked, but as stated in Section 1.2.1, it is not possible to evaluate every solution.

Additionally, the LS algorithm is endowed with a termination criterion [19, p. 48] which stops the search on newly generated neighbourhoods after N_c steps of no improvement[3].

In terms of the original problem, the algorithm can be adapted by extending the neighbourhood definition Γ to include changes in the lamellas orientation. Therefore, Γ around the current solution s is generated by performing every possible swap of two lamellas and flipping every lamella. Hence $|\Gamma| = \frac{(n_l-1)\cdot n_l}{2} + n_l$.

In contrast to the simplified problem Equation (4.1) can directly be used as objective function.

4.5 Implementation of iterated local search

As described in Section 3.2.1, LS can be enhanced by applying perturbations to generate new not yet visited neighbourhoods.

A reasonable definition of the neighbourhood is as follows: The LS algorithms are bound to one beam only and the perturbation is applied in form of an interchange of lamellas between two distinct beams. This implies that the LS operates on a smaller neighbourhood than the one proposed in Section 5.2. Figure 4.4 shows an overview of the described algorithm.

For the stated problem, n_b different LS computations are employed in an inner loop, where each is defined as follows: Let the objective function for LS_i – based on Equation (4.5) – be defined as

$$f(s_i) = (EI_b(s_i))^{-1} \tag{4.16}$$

and let the neighbourhood Γ_i around the current solution s_i be generated by performing every possible swap of two lamellas within s_i. Therefore, $|\Gamma_i| = \frac{(n_{b,l}-1)\cdot n_{b,l}}{2}$.

Each LS_i run is endowed with a convergence criterion which stops the search on Γ_i after N_c steps of no improvement. This guarantees the optimal layout for each beam, given the corresponding set of lamellas.

After all n_b LS computations have converged, the worst solution denoted by s_k and a randomly chosen solution denoted by s_m, $s_m \neq s_k$ are picked.

Subsequently, the perturbation is implemented by defining a new LS which searches in the neighbourhood $\Gamma' = L_k \times L_m$ where L_k denotes the lamellas used in solution s_k and L_m denotes the lamellas used in solution s_m. For the pair $p_i \in \Gamma'$ of lamellas to be swapped between the

[3] In detail this criterion is implemented by comparing the maximum value of the last N_c values with the minimum value of the last N_c values. The comparison is made in an approximate manner for float numbers where half of the significant digits are compared.

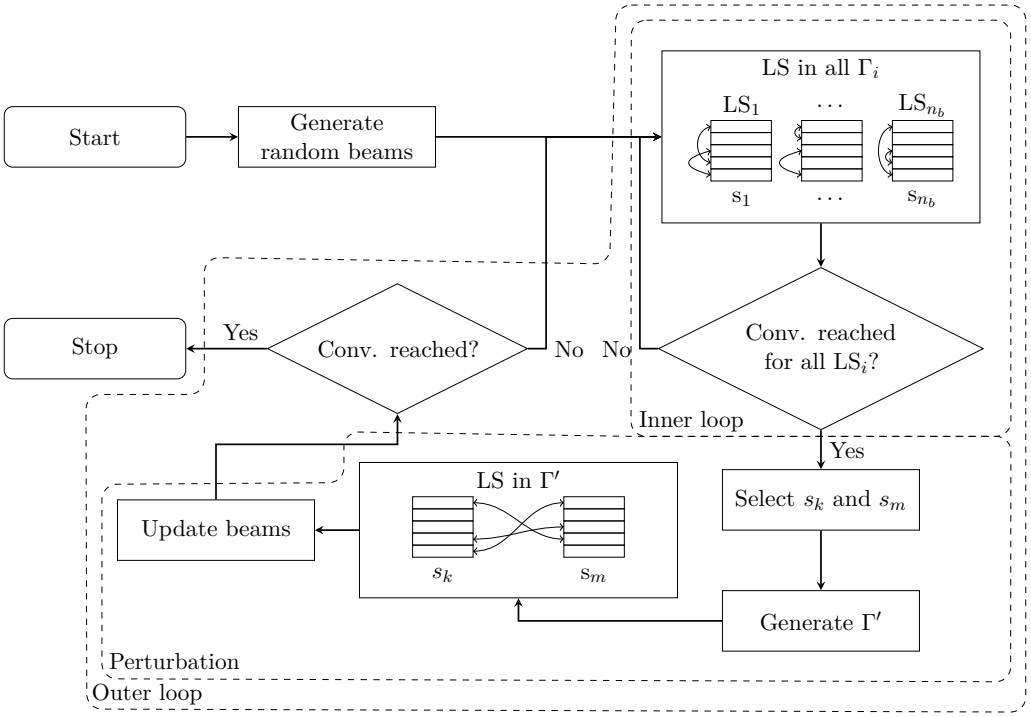


Fig. 4.4: Overview of the implemented ILS algorithm.

solutions s_k and s_m two new solutions $s'_{k,i}$ and $s'_{m,i}$ are generated. The corresponding objective function f' is defined as

$$f' : \begin{cases} 0, & \text{if } f(s'_{k,i}) < f(s_k) \text{ and } f(s'_{m,i}) < f(s_k) \\ 1, & \text{if } s'_{k,i} = s_k \text{ and } s'_{m,i} = s_m \\ 2, & \text{otherwise.} \end{cases} \tag{4.17}$$

This leads to the effect that a perturbation is only accepted if the newly generated beams are better than the previously worst beam s_k. Consequently, in case $f' = 0$, the resulting beams $s'^{*}_{k,i}$ and $s'^{*}_{m,i}$ replace s_k and s_m respectively. For this LS procedure, no convergence criterion is needed, as the search stops immediately after finding a pair p_i which satisfies $f' = 0$.

As a convergence criterion, for ultimately stopping the outer loop, the coefficient of variation c_v is used which decreases as all beams become similar. Therefore, further LS on Γ' will not lead to any improvement. The criterion can be formulated as $c_v[f(s_i)|i \in \{1, \ldots, n_b\}] \leq \epsilon_c$.

As well as LS, ILS can be adapted to the original problem by extending the neighbourhood definition.

For this implementation the neighbourhood Γ_i around the solution s_i is extended – in addition to performing every possible swap – by flipping every lamella within s_i. Therefore, $|\Gamma_i| = \frac{(n_{b,l}-1)\cdot n_{b,l}}{2} + n_{b,l}$. As the purpose of the neighbourhood Γ' is the exchange of lamellas between different solutions, there is no need for extending this definition as during this swap the orientation of both lamellas remains unchanged.

The objective function for ILS can be defined based on Equation (4.1) as

$$f(s_i) = q_{\max}(s_i). \tag{4.18}$$

Furthermore, the objective function for performing swaps between beams can be inherited, as it is defined in terms of Equation (4.18).

4.6 Implementation of tabu search

As described in Section 3.2.2, the foundation of TS are tabus and aspiration criteria. The tabus for the stated problem are defined as follows:

1. Do not reverse the previous swap, i.e. do not swap one of the involved lamellas back to the previous position.

2. Do not move one of the involved lamellas to the beam they came from.

Every tabu is endowed with an adapted tabu tenure $N_{T,i}$. As the first tabu is less strict than the second one, the tabu tenure $N_{T,1}$ is higher than the tabu tenure $N_{T,2}$.

The aspiration criteria is implemented in form of accepting solutions which are better than the previously known best solution.

Similar to the LS implementation described in Section 4.4, TS is endowed with a convergence criterion to stop after N_c steps of no improvement and a heuristic component to reduce the search space. In case of TS, the heuristic used for LS is not applicable, as the algorithm at first needs to gain knowledge about the objective function values within the neighbourhood. Gendreau et al. [19] proposed a solution to only evaluate a randomized portion Γ', $|\Gamma'| = N_n$ of the neighbourhood Γ around s to reduce the computation amounts.

For the original problem, the above tabu definition is simply extended by

3. Do not flip a previously flipped lamella back.

The comparison of lamellas for tabus 1 and 2 is performed by neglecting the orientation of the lamella. Thus, it is not necessary to adapt these tabus, as they cannot be tricked by flipping a lamella i.e. it is not possible that a lamella is moved back to a beam (tabu 2) or back to a position (tabu 1) after it was flipped.

The aspiration and the convergence criterion are inherited from the definition for the simplified problem.

The neighbourhood Γ around the current solution s is generated similar to LS, described in Section 4.4, however the heuristic component needs to be adapted as described above.

4.7 Implementation of tabu search using candidate list solutions

An alternative solution, proposed by Gendreau et al. [19], to reduce the number of needed evaluations is to generate a useful subset Γ' of Γ by means of candidate list strategies. The

implementation of ILS in Section 4.5 is based on the same principle, as swaps between beams are explicitly made for the worst and one of the other beams. Therefore, the list of candidates of swaps is reduced to a smaller subset of the search space, which has a higher chance of improving the worst beam. In terms of a TS implementation this can be realized by redefining the neighbourhood Γ around s to a neighbourhood Γ' around s which favours moves from weak to strong beams. In this implementation Γ' is defined by:

1. Allowing repositioning of lamellas only between the worst beam and one of the others and
2. allowing repositioning of lamellas within arbitrary beams.

The tabus for movement 1 can be inherited from the initial tabu search implementation. For movement 2 only tabu 1 is needed, as the swap is performed only within a single beam.

In context of the original problem, the neighbourhood Γ' around s is extended by

3. Allow flips of lamellas.

The tabus for movement 1 (between beams) and movement 2 (within beams) can be inherited from the simplified problem, however – as described in Section 4.6 – comparison must be performed solely based on the lamella ID. The tabu for the newly added movement 3 equals the one for the simplified problem.

4.8 Implementation of genetic algorithms

Within the GA approach, Equation (4.6) can be used as objective function or – in terms of GAs – as fitness function which shall be maximised.

In contrast the objective function for the original problem needs to me minimised, hence as $q_{\max}(b_i) \neq 0 \,\forall\, b_i$ the fitness function can be expressed as

$$\min \left\{ \frac{1}{q_{\max}(b_i)} | b_i \in \Phi(\pi^*) \right\}. \tag{4.19}$$

The GA is mainly implemented based on the procedures laid out in Goldberg [21], Reeves [44], Baker [2], Blickle et al. [4], Fox et al. [18] and Larranaga et al. [31]. Numerous selection-, crossover- and mutation methods are implemented, which will be described in the following section. Generally it should be noted that, there is no preference of any of the following described methods as their behaviour and effect on the optimisation procedure varies from problem to problem. Thus, the following sections are presented in a general manner, details on the actual applied operations are given in Section 5.8 and Section 6.3.8.

4.8.1 Selection

In the selection stage, the GA procedure aims at mimicking the concept of survival of the fittest. Consequently, individuals are assigned a fitness value, reflecting its performance. The selection is then implemented such that it trends to select fitter individuals over less-fit ones. In the following the employed selection procedures are discussed.

Roulette wheel selection is implemented based on the description by Goldberg [21]. This algorithm selects chromosomes by virtually spinning a roulette wheel with slots sized according to the chromosomes fitness f_i, as shown in Figure 4.5a. The number of spins depends on the desired number of members in the mating pool. In this implementation the population size is assumed to be constant during the optimisation procedure, thus the number of spins equals the number of individuals in the population, denoted by N_p. Furthermore, this selection approach allows for multiple copies of the same individual in the mating pool.

The slot size or probability $P[i]$ a member i is part of the mating pool is determined by

$$P[i] = \frac{f_i}{\sum_{i=1}^{N_p} f_i}, \tag{4.20}$$

where f_i denotes the individual's fitness.

This form of selection is insofar problematic as in an early stage, where the average fitness value is far from the maximum fitness value, chromosomes with higher fitnesses are selected disproportionately more often and therefore the algorithm tends to converge against those members of the population and loose its exploratory capabilities. On the other hand, in a later stage the populations mean fitness value could already be close to the maximum value, while showing significant differences between the population members. By using a roulette wheel selection in this stage $P[i]$ would be almost identical for all N_p members, as the absolute difference between the values for f_i is small. Therefore, the resulting roulette wheel is not biased for fitter population members, thus resulting in an entirely random selection.

Goldberg [21] mentions fitness scaling as a useful procedure to improve both the early stage and the later stage. An approach is to perform linear scaling, which requires a relationship between the raw fitness f and the scaled fitness f' as follows:

$$f' = a \cdot f + b. \tag{4.21}$$

Further, to ensure that average members remain average members by means of the raw fitness and the scaled fitness, the following relation must hold:

$$f'_{avg} = f_{avg} \tag{4.22}$$

The scaling is applied by prescribing the probability of the best member being part of the mating pool to be relative to the average fitness. This relation can be formulated in terms of the scaling factor C_{mult} as follows:

$$f'_{max} = C_{mult} \cdot f_{avg}. \tag{4.23}$$

By specializing Equation (4.21) for Equation (4.22) and Equation (4.23), a and b can be expressed in terms of the raw fitness and C_{mult} as

$$a = (C_{mult} - 1) \cdot \frac{f_{avg}}{(f_{max} - f_{avg})} \qquad \text{and} \qquad b = (1 - a) \cdot f_{avg}. \tag{4.24}$$

As Equation (4.21) can yield negative values for under average members it needs to be redefined as:

$$f' = \begin{cases} a \cdot f + b, & b > -a \cdot f \\ 0, & \text{otherwise} \end{cases} \tag{4.25}$$

According to Goldberg [21], for population sizes between 50 and 100 members, C_{mult} typically ranges from 1.2 to 2.

Reeves [44] states that roulette wheel selection is problematic as the actual number of times a chromosome is selected may be very different from its expected value. Thus, being subject to stochastic effects.

Stochastic universal selection (SUS) is implemented based on the description by Baker [2]. SUS is capable of overcoming the stochastic variability that affects roulette wheel selection. Baker [2] uses following terms to assess selection methods for GAs:

Bias: Bias is defined as the absolute difference between the expected number of copies of a chromosome in the mating pool and the actual number of copies. Ideally zero bias should be achieved to reduce the influence of stochastic effects.

Spread: Spread is defined as the range of actual number of copies a chromosome has in the mating pool. The minimum spread is defined as the spread that permits zeros bias. For a chromosome i the minimum spread can be expressed as

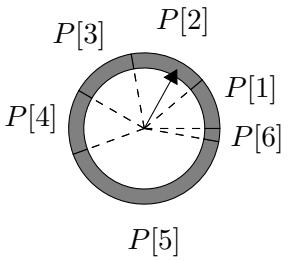
$$\{\lfloor E[i] \rfloor, \lceil E[i] \rceil\}, \tag{4.26}$$

where $\lfloor \bullet \rfloor$ is the greatest integer less than $\bullet$ and $\lceil \bullet \rceil$ is the smallest integer greater than $\bullet$.

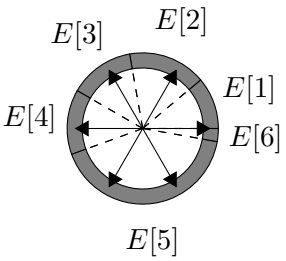
To reuse the analogy of the roulette wheel one can imagine SUS as a roulette wheel with N_p equally spaced arms where the slot size equals the expected value of the corresponding individual. A single spin results in N_p selected chromosomes. As the N_p arms are equally spaced and the sum of the slot sizes equals N_p a chromosome i is guaranteed to have at least $\lfloor E[i] \rfloor$ and not more than $\lceil E[i] \rceil$ copies in the mating pool. Therefore, SUS has minimum spread. This is obvious when looking at Figure 4.5b, as it is impossible for e.g. individual 6 to have more than one copy in the mating pool whereas individual 5 always ends up with at least 2 copies. Furthermore, SUS is zero biased, as in a randomly ordered population the outcome is solely based on the initial spin and the slot size on the roulette wheel.

As the mating pool resulting from SUS is grouped by individuals, it is inevitable to shuffle the resulting list in order to prevent the following crossover operation from crossing two equal individuals.

Linear ranking is implemented based on the description by Reeves [44]. In the case of linear ranking it is assumed, that the probability of an individual being selected can be determined



(a) Basic roulette selection for 6 individuals



(b) SUS for 6 individuals with a 6-armed spinner

Fig. 4.5: Comparison of roulette wheel selection and SUS using a roulette wheel

solely by ranking according to the individuals fitness in an descending order. This assumption causes some loss off information, as the actual distribution of the fitness values is omitted, but it allows to formulate a simpler selection method, which is independent of rescaling the fitness values, as described for roulette selection. The probability $P[k]$ that an individual ranked k is selected is defined by:

$$P[k] = a + b \cdot k, \tag{4.27}$$

where a and b are constant scalars.

As $P[k]$ is a probability distribution the condition

$$\sum_{k=1}^{N_p} (a + b \cdot k) = 1 \tag{4.28}$$

must be satisfied.

Equation (4.28) can be rewritten in from of an arithmetic series for the N_p-th element as

$$a \cdot N_p + b \cdot \frac{N_p \cdot (N_p + 1)}{2} = 1. \tag{4.29}$$

Reeves [44] introduces the selection pressure Ψ, which expresses the desired ratio between the probabilities that the best individual is contained in the mating pool and the median individual is contained in the mating pool as follows:

$$\Psi = \frac{P[\text{best}]}{P[\text{median}]}, \tag{4.30}$$

The selection pressure can be specialized for the case that N_p is even in form of

$$\Psi = \frac{a + b \cdot N_p}{a + b \cdot \frac{N_p}{2}} \tag{4.31}$$

and for the case that N_p is odd in form of

$$\Psi = \frac{a + b \cdot N_p}{a + b \cdot \frac{N_p + 1}{2}}. \tag{4.32}$$

From Equation (4.29), Equation (4.31) and Equation (4.32), definitions for a and b can be derived.

The even case is defined as

$$a = \frac{2 - \Psi}{M + \Psi - 1} \qquad \text{and} \qquad b = \frac{2 \cdot (\Psi - 1)}{M \cdot (M + \Psi - 1)}, \tag{4.33}$$

the odd case is defined as

$$a = \frac{2 \cdot M - \Psi \cdot (M + 1)}{M \cdot (M - 1)} \qquad \text{and} \qquad b = \frac{2 \cdot (\Psi - 1)}{M \cdot (M - 1)}, \tag{4.34}$$

where $1 \leq \Psi \leq 2$.

Tournament selection is implemented as described by Reeves [44] and Blickle et al. [4]. Tournament selection is a selection method where a set of t individuals is selected randomly from the population and compared. The best is chosen to be added to the mating pool. This process is repeated until the desired number of elements needed to form the next generation is obtained.

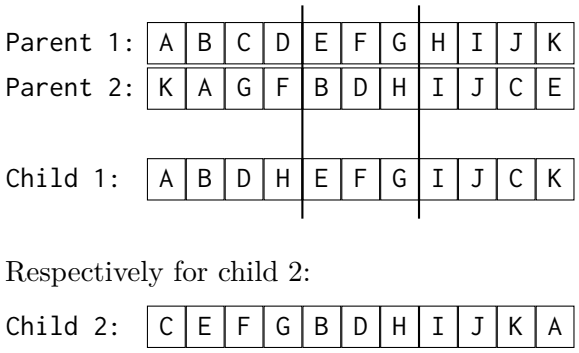
Generally the algorithm is very efficient because the population does not need to be sorted beforehand, but it has similar downsides as roulette wheel selection as it is also subject to stochastic effects due to randomly selecting the subset of t individuals.

4.8.2 Crossover

During crossover the individuals, selected in the selection phase, are combined to form the subsequent generation. Thereby mainly characteristics of strong individuals are inherited as, due to the fitness based selection, the probability a former strong individual is involved in a crossover is higher. As the stated problem is of combinatorial nature it is important that the GA uses crossover algorithms which preserve this nature. That is, each gene (representing a individual lamella) can only occur once in each chromosome. Therefore, usually after a phase of copying parts of the parent chromosomes to the child chromosome a reconstruction phase is performed which re-establishes a correct sequence.

Ordered crossover (OX) is implemented as described by Fox et al. [18]. The idea behind OX is to preserve the order of a subsequence of genes from one parent and the relative order of the remaining genes from the other parent. The subsequence is defined by two random cuts and is moved to the same position within the child. Starting from the second cut, the remaining genes from the other parent are moved to the child, skipping every gene contained in the previously moved subsequence. The second child can be generated in the same manner by swapping the parents. Figure 4.6 shows an example for OX.

Cycle crossover (CX) is implemented as described by Fox et al. [18]. CX is based on the principle that every position of a gene within the child is based on the position the same gene is located within one of the parents. This is done by searching for a sequences of genes which can



1. Move E,F,G from parent 1 to child 1.
2. Move I,J,C from parent 2 to child 1.
3. Skip E as it was already moved in step 1.
4. Move K from parent 2 to child 1
5. Move A from parent 2 to child 1
6. Skip G,F as both were already moved in step 1.
7. Move B,D,H from parent 2 to child 1

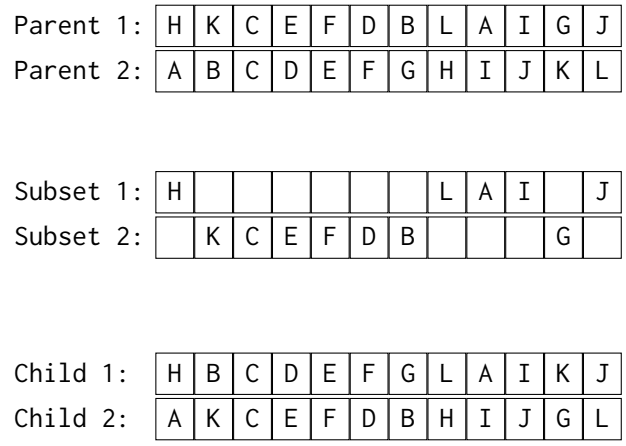
Fig. 4.6: Example for OX [18, p. 286] including the steps needed for generating child 1

be moved to a child without interfering with the genes at the remaining positions of the other parent. Thus, one child consists of two subsets of parent genes where each subset is disjoint from the positions of the second subset. The child can then be constructed by combining the two subsets.

The subsets can be generated by first selecting a single gene k from parent 1 which is used as a seed for the following operations. Then the conflicting location within parent 2 i.e. the position X where the gene with the same value as gene k is located is obtained and the gene l from parent 1 at the conflicting position X is used as the next gene k which is copied to the child from parent 1. This process is repeated until a position X is reached which has already been copied, thus the cycle is closed and the remaining genes from parent 2 can be copied to the child. Figure 4.7 shows an example for CX including both subset extracted from parent 1.

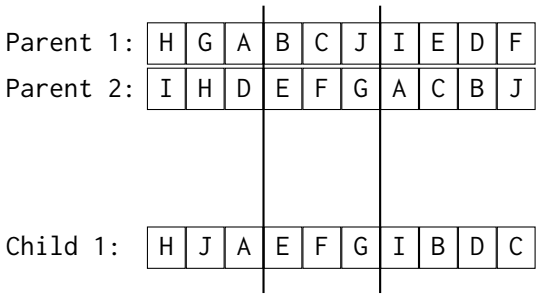
Partially mapped crossover (PMX) is implemented as described by Fox et al. [18] and Larranaga et al. [31, p. 138]. PMX uses, equally to OX, a subsequence of genes defined by two cuts from one parent and copies them to one child. The remaining places are filled by respecting the order and the position of the remaining genes from the other parent. This is implemented by at first copying the defined subset from parent 1 to the child. Then the genes within the subsets from parent 1 and parent 2 are explicitly mapped, as shown in Figure 4.8. As all mappings have been declared the genes from parent 2 can be copied to the child, in case one copy violates the nature of the CO problem the corresponding mapping is used to replace the gene.

Larranaga et al. [31] states that it is possible that mappings could build a chain of mappings. This happens if one mapping targets a gene within the subsequence e.g. the subsequences A,D,F and D,C,G result in the mappings $A \leftrightarrow D$, $D \leftrightarrow C$ and $F \leftrightarrow G$. When the algorithm requests the mapping for the gene to replace A with, the first occurring mapping is $A \leftrightarrow D$. But as D is contained in the subsequence, it is not suitable for a replacement as it violates the nature of the CO problem. Thus, the mapping $D \leftrightarrow C$ is used and C is added to the child, in place of A



1. Generate subset 1
   a) Pick [H] from parent 1 as a starting point. [H] replaces [A] from parent 2.
   b) Pick [A] from parent 1 to compensate the replaced [A] from parent 2. [A] replaces [I] from parent 2.
   c) Pick [I] ...
   d) Pick [L] from parent 1 to compensate the replaced [L] from parent 2. [L] replaces [H] thus the cycle is closed.
2. Generate subset 2 by picking the position not contained in subset 1 from parent 1.
3. Move subset 1 to child 1
4. Fill the remaining empty position in child 1 with the genes at the matching position from parent 2.
5. Move subset 2 to child 2
6. Fill the remaining empty position in child 2 with the genes at the matching position from parent 2.

Fig. 4.7: Example for CX [18, p. 287] including the generation of both children



1. Define mappings for [B,C,J] from parent 1 and [E,F,G] from parent 2: $B \leftrightarrow E$, $C \leftrightarrow F$ and $J \leftrightarrow G$.
2. Move [E,F,G] from parent 2 to child 1.
3. Move [I] , [D] , [H] and [A] from parent 1 directly to child 1, as no mapping restricts these movements.
4. Put [B] into child 1 at the position of [E] in parent 1 ($B \leftrightarrow E$).
5. Put [C] into child 1 at the position of [F] in parent 1 ($C \leftrightarrow F$).
6. Put [J] into child 1 at the position of [G] in parent 1 ($J \leftrightarrow G$).

Respectively for child 2:

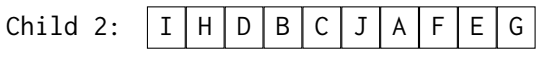


Fig. 4.8: Example for PMX [18, p. 287]

4.8.3 Mutation

In the mutation phase the GA tries to maintain the diversity of the population by applying small random changes to the population generated by crossover. Similar to the crossover operators described in Section 4.8.2, mutation needs to preserve the combinatorial nature of the problem. Therefore, mutation is applied in form of

1. swapping two lamellas or
2. replacing an entire individual with a random beam setup.

Larranaga et al. [31] refers to the first method as exchange mutation or swap mutation. The mutation is performed by selecting two positions within the chromosome at random and performing a swap. This form of mutation is sufficient for the simplified problem described in Section 4.2. In order to perform a useful mutation for the original problem, the algorithm is further extended by a flip operation which changes the orientation of a lamella. The applied mutation method is picked randomly by equal chances.

4.8.4 Elitism

Another concept borrowed from evolutionary theory is elitism, which describes the survival of the fittest individuals over multiple generations. It is implemented as described by Reeves [44], that is ϵ_c of the fittest individuals are moved directly to the next population. This method assures that the best performing solutions are not destroyed by crossover and mutations. The size of the mating pool is determined by $N_p - \epsilon_c$.

4.8.5 Chromosome repair

Reeves [45] addresses the situation of different permutations actually representing same solutions. It was already mention in Section 1.2, that the order in which the beams occur within a chromosome, has no effect on the actual solution.

The crossover operations from Section 4.8.2 are designed to generate offspring which is based on its parents. Therefore, when combining two equal chromosomes the generated offspring must be equal as well. From Figure 4.9 it can be seen that although chromosomes 1 and 2 represent the same solution for $n_b = 3$ and $n_{b,l} = 4$ (as they consist of equal beams), after applying PMX, two new solutions are created.

The repair process is performed by sorting the beams within the chromosome according to the number of the lamella at the first location within the beam. The sorting process is applied directly after the mutation was performed. Figure 4.10 shows the same problem definition with repaired chromosomes. As expected the generated offspring now matches the parents.

4.8.6 Initial population

Multiple sources [44, 46, 49] address the generation of the initial population. The aim is to obtain a diversified initial population, which covers the search space adequately. Otherwise

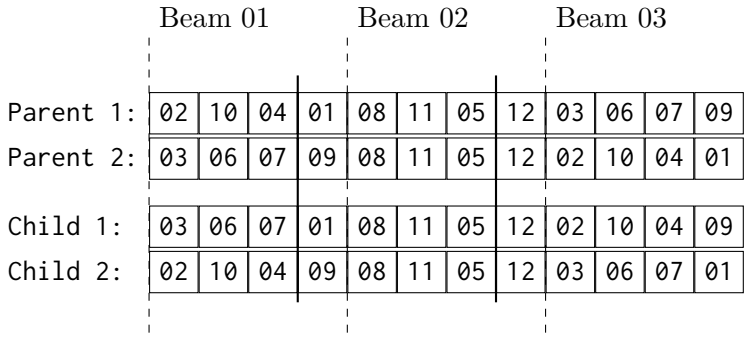


Fig. 4.9: Example for not repaired chromosome 1 and 2 representing equal solution.

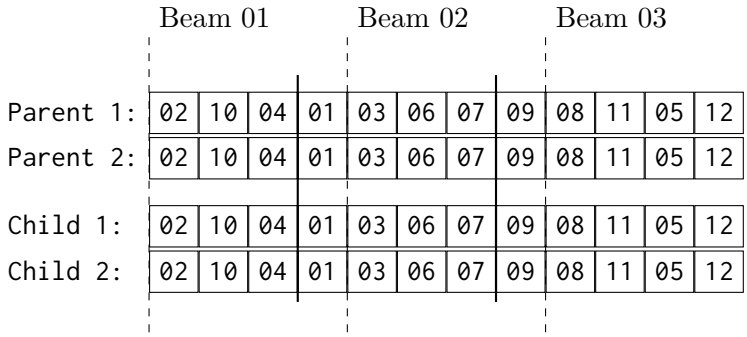


Fig. 4.10: Example for repaired chromosome 1 and 2 representing an equal solution.

an unsuitable initial population could cause premature convergence or reduce the exploratory capabilities of the GA.

Reeves [46] proposed the principle that every solution within the search space should be reachable by crossover only. This implies that every allele must be at each locus at least once within the population. Based on this principle, Reeves [46] derived an equation to calculate the probability that every allele is present at every locus based on the population size, the chromosome length and the number of alleles.

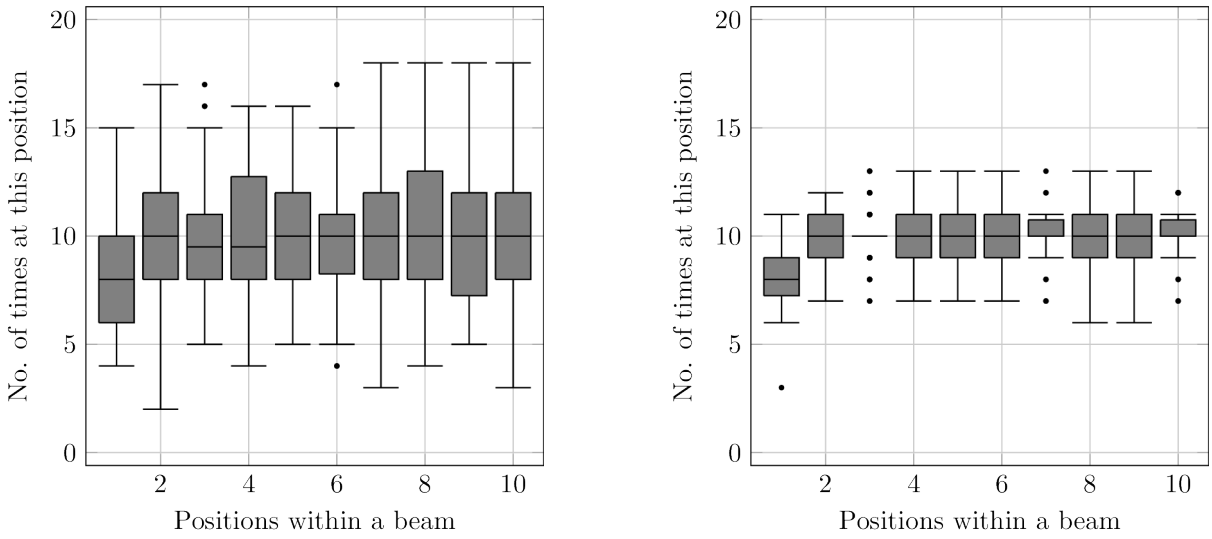
This approach apparently works for problems with small number of alleles, but for larger numbers the minimum population size tends to be too large to be usable [46]. Furthermore, the used principle does not hold for CO problems, since herein, as described in Section 4.8.2, the used crossover operations incorporate swapping operations and therefore are able to reach solutions with alleles at positions which were not present in the initial population.

Coelho et al. [7], Reeves [44], and Talbi [49] propose the usage of latin hypercube sampling to generate a pseudo random initial population. A latin hypercube sample j for discrete variables can be obtained by generating random permutations π_i of all alleles for every gene i and using the gene values $\pi_i(j)$ for the solution j. For CO problems, where the genes cannot be sampled independently, latin hypercube sampling is not applicable as well. Therefore, the method used in this work is a modified implementation of the so-called algorithm P used for shuffling, introduced by Knuth [30, p. 139]. The algorithm picks an entry at a random position within the vector which shall be shuffled and swaps it with the element at the last position. The element, which is

now at the last position, is at its final location within the shuffled vector, thus the next randomly chosen element is swapped with the second last element. This process is repeated until every element was swapped.

Herein, the algorithm is modified such that lamellas are evenly distributed over all available positions. In detail, a randomly chosen position is rejected in case the lamella was used an above-average amount of times at this position.

Figure 4.11 shows a comparison for $n_b = 5$, $n_{b,l} = 10$, and $N_p = 100$ of a purely random shuffle shown in Figure 4.11a and the adapted pseudo random version shown in Figure 4.11b, where for every possible location within a beam the number of times a lamella was used at this location is plotted.



(a) Pure random shuffle **(b)** Pseudo random shuffle

Fig. 4.11: Number of lamellas at locations within beams. Location 1 corresponds to the top location and location 10 to the bottom location.

Figure 4.11 clearly shows that the modified algorithm generates a more homogeneous distribution of the lamellas through the beams. This approach of generating the initial population is insofar advantageous, as it is not possible that a lamella only occurs at certain locations within the beams. Thus, the starting solution is guaranteed to have a broader overview of the solution space than a starting solution generated at random.

The algorithm is further extended for generating an initial population respecting the orientation of a lamella. This is implemented based on the representation scheme proposed in Equation (4.15). The initial vector, which shall be shuffled is build solely from not flipped lamellas. After one position was picked randomly, the lamella associated with this position gets a random orientation assigned i.e 1 or 0 is added to the lamella ID. As the ID now contains the orientation, it is compared against the times it was used at the desired position and is rejected in case of an above-average amount of times.

Chapter 5

Application of metaheuristic algorithms for the simplified problem

5.1 Practical example

In the following numerous example calculations are performed and optimisation approaches are discussed to verify or support certain ideas. Thereby the values for n_l, n_b and $n_{l,b}$ are set to:

$$\begin{aligned} n_l &= 50, \\ n_b &= 5, \text{ and} \\ n_{l,b} &= 10. \end{aligned}$$

The lamella data used in this test sample comprises the lamellas 59–109 (except lamella 71 due to missing data of $E(x)$) of the grading class LS22, from the experiments in [47].

The GLT beam test setup is shown in Figure 5.1. The dimensions and loads used for the present example are:

$$\begin{aligned} a_1 &= 1.38 \text{ m} \\ a_2 &= 1.38 \text{ m} \\ a_3 &= 0.63 \text{ m} \\ F &= 5.0 \text{ kN} \end{aligned}$$

The lamellas are assumed to have equal cross sectional dimensions of $l_h = 0.033\,\text{m}$ in height and $w = 0.09\,\text{m}$ in width.

5.2 Application of local search

The LS is at first performed starting from a single random beam setup to gain insight into the algorithms behaviour. The progress of the optimisation procedure is shown in Figure 5.2. Obviously the algorithm converged against a solution, however it is not clear what the actual best solution is, thus the performance of one algorithm is assessed in comparison to the other

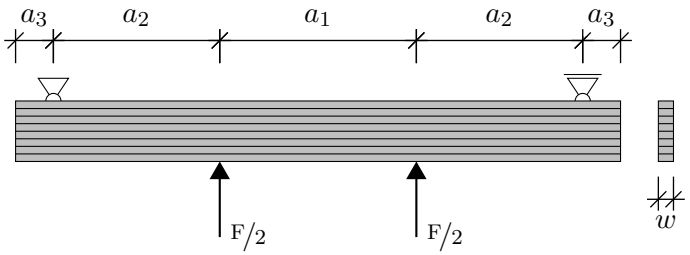


Fig. 5.1: Definition of dimensions for the test setup considered

discussed algorithms. Furthermore, the actual improvement entirely depends on the quality of the initial solution, thus it is not used as an criterion for measuring the performance of the used algorithms. Details on the possible range of improvement are given in Section 6.5.1.

The optimisation shown in Figure 5.2 was performed in about 3 s on a 2015 MacBook Pro with a 3.1 GHz Dual-Core Intel i7 CPU and 16 GB RAM. The time needed for the optimisation is, of course, only a rough estimate of the average time needed for the optimisation, as it is strongly depended on the influence of background processes. Nevertheless, it provides the reader a first impression of the duration of the discussed procedures.

For more complex objective functions, the overhead of the computation time spent on the algorithm can be neglected in relation to the computation time needed for evaluation of the objective function. Therefore, in order to provide a basis for comparing different algorithms, the progress is always plotted in comparison to the number of distinct calls[1] to the objective function. Thus, eliminating the variability involved when actually comparing the computation time.

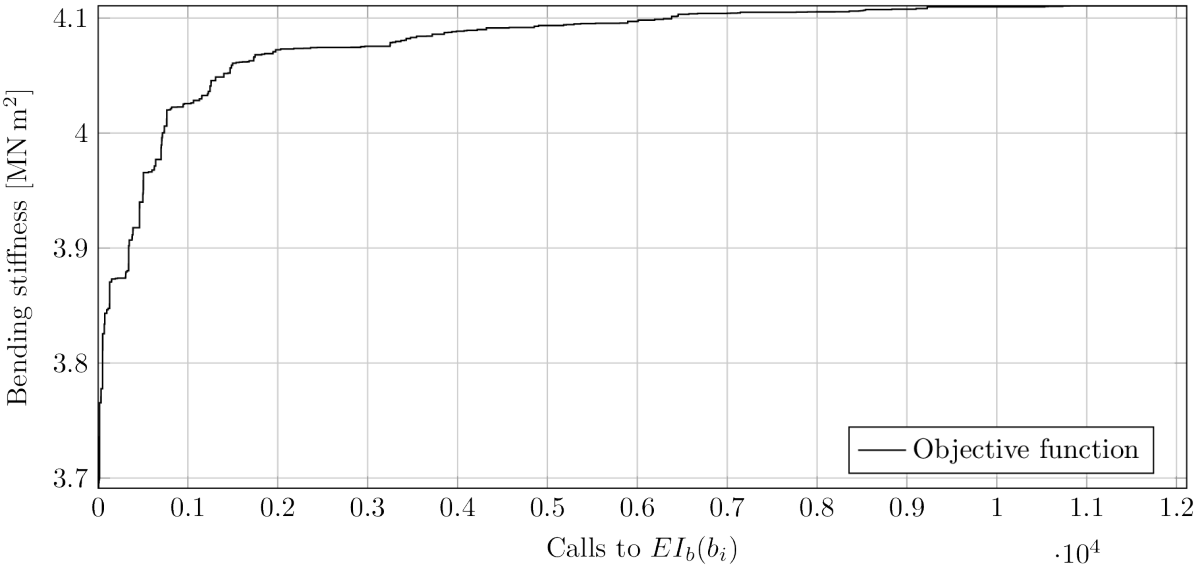


Fig. 5.2: Exemplary optimisation progress of the LS algorithm starting with a random beam setup for $N_c = 50$.

[1]The reasoning for capturing only distinct functions calls is further explained in Section 6.1

5.3 Application of iterated local search

Similar to LS, the ILS algorithms is tested starting from a random initial beam setup. Figure 5.3 shows that ILS converged to a similar value as LS. The algorithm needed about 2 s to performed the optimisation task. When comparing the bending stiffness of a single beam to the value of the objective function of the optimisation problem, the effect of exchanging lamellas between beams is recognizable, as an exchange of lamellas on the one hand weakens a strong beam and on the other hand improves a weak beam. Particularly exchanges between the currently strongest beam and the weakest beam can be clearly seen in the progress curves.

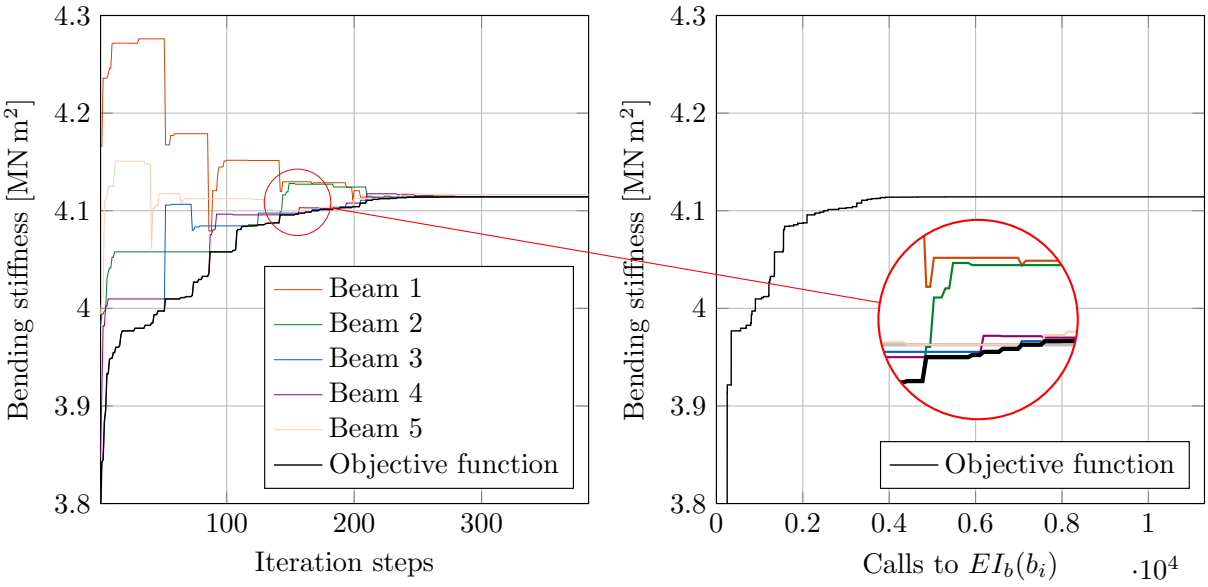


Fig. 5.3: Exemplary optimisation progress of a ILS run for $N_c = 25$ and $\epsilon_c = 0.0001$.

5.4 Comparison of local search and iterated local search

For comparison of LS and ILS both algorithms are tested on 1000 randomly chosen beam setups. The results of these runs are shown in Figure 5.4. The black line within the shaded area is the 5 % trimmed mean of the objective function, i.e. for computation of the mean value curve, the highest 5 % and the lowest 5 % of values are neglected. The reason for evaluating the trimmed mean is that it is less sensitive to outliers while still preserving a reasonable estimate of the mean value. The shaded area spans the distance from the first to the third quartile and the grey outer lines mark the minimum and the maximum values of the objective function.

From Figure 5.4 the following conclusion can be drawn:

1. The shaded areas marking the inner quartile range show that the 1000 runs are narrowly distributed around the mean value curves. While in the overlapping areas of LS and ILS any statement has to be carefully evaluated, in the remaining parts, significant differences for both mean value curves are observed.

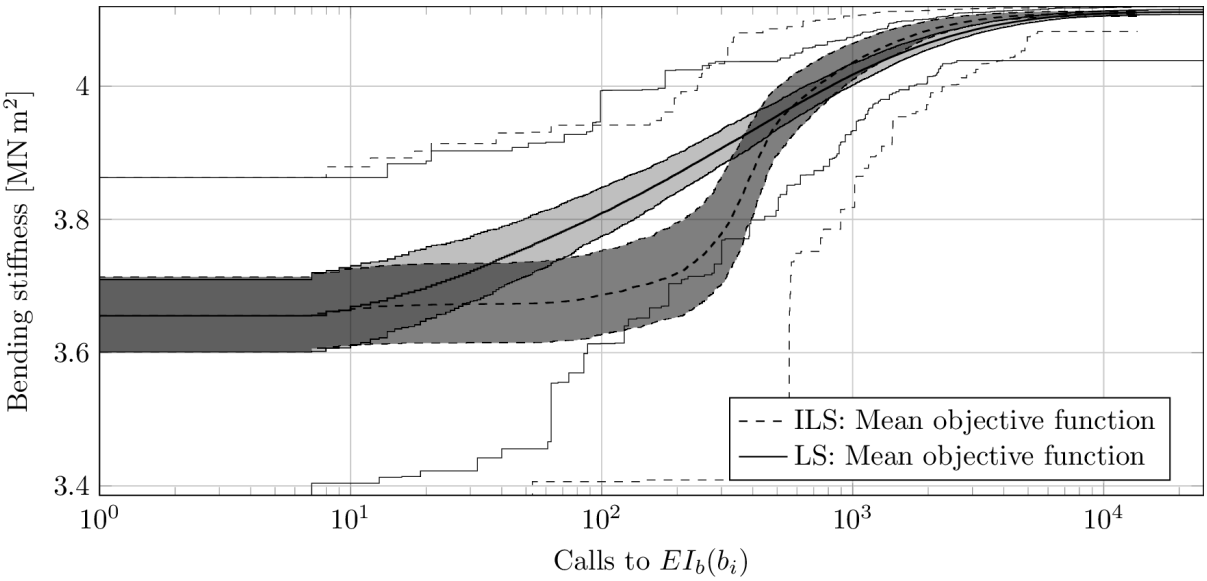


Fig. 5.4: Comparison of 1000 LS and ILS runs starting from equal beam setups.

2. ILS initially needs more computations to improve the starting solution. This could be caused by the initial task of ILS of improving every single beam, as the worst beam – defining the value of the objective function – consists of mainly weak lamellas, thus the potential range of improvement is low. Nevertheless, at the point where ILS mainly improves the current solution by explicitly moving lamellas from superior beams to inferior ones, the improvement rate rapidly increases and ultimately ILS converges faster than LS.

3. LS gets stuck at local optima more often than ILS which can be seen by looking at the maximum value path (which in this case is the lower grey solid curve) in Figure 5.4, as ILS is "trapped" at a later stage than LS

5.5 Application of tabu search

TS is initially performed on a random beam setup. The progress of the optimisation procedure for the run for $N_c = 100$, $N_n = 200$, $N_{T,1} = 50$ and $N_{T,2} = 10$ is shown in Figure 5.5. Although a heuristic component is used, the number of evaluations of Equation (4.5) compared to the ones needed by LS or ILS is larger by about a factor of 4. As a consequence TS spent about 41 s of computation time on optimising the initial beam setup. The solution the algorithms converged to, is similar to the solutions obtained by the other algorithms.

5.6 Application of tabu search using candidate list strategies

The extension of TS by using a neighbourhood generated by means of candidate list strategies, should have a similar effect as the LS adaption ILS. In contrast to ILS, TS using candidate list solution never actually operates on a single beam, nevertheless the progress of improving single

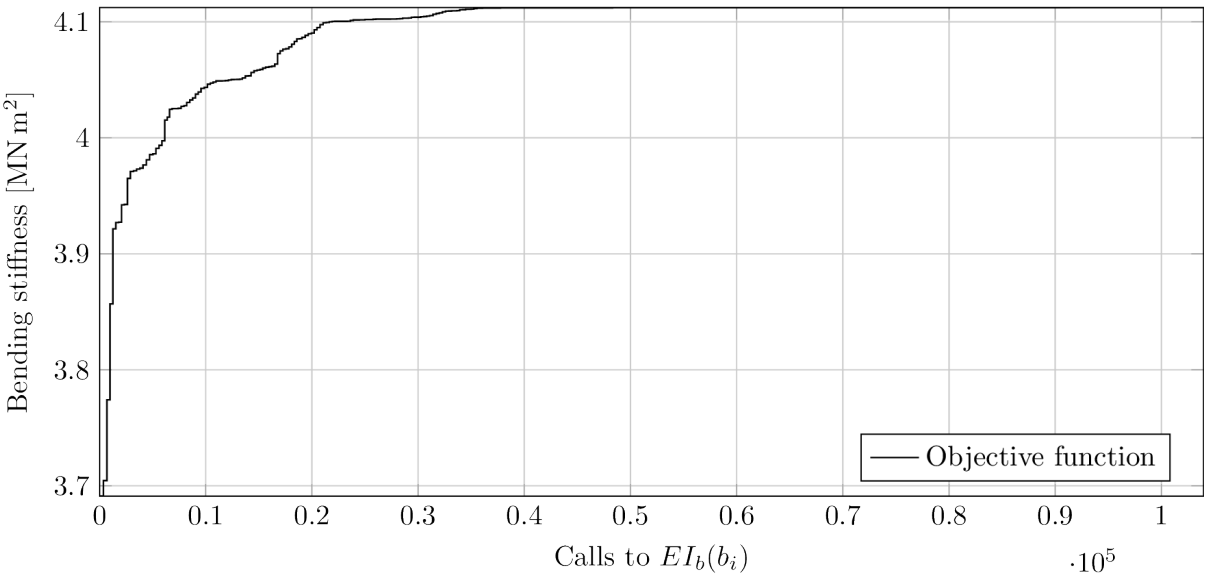


Fig. 5.5: Exemplary optimisation progress of a TS algorithm starting with a random beam setup.

beams can be calculated by evaluating Equation (4.5) for every beam contained in the current best solution. The progress for $N_c = 100$, $N_n = 100$, $N_{T,1} = 50$ and $N_{T,2} = 10$ is shown in Figure 5.6. Similar to the progress of ILS, the effect of deteriorating a strong beam to improve the weakest beam is recognizable in Figure 5.6. The time needed for the optimisation was about 6 s. This reduction in comparison to the original TS implementation is also reflected in a comparably fast convergence.

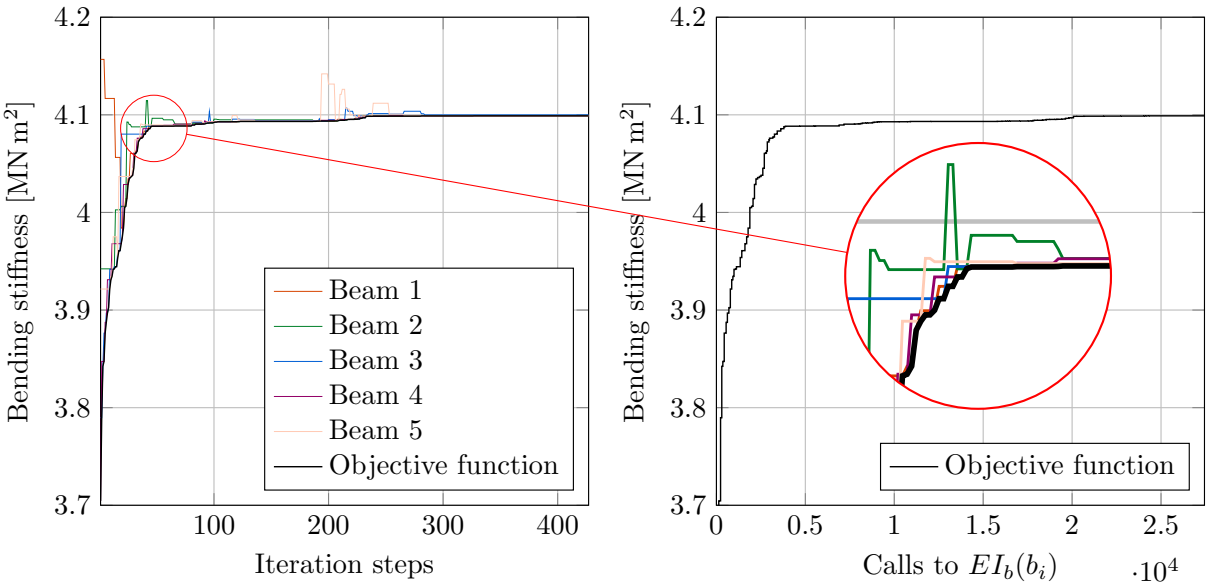


Fig. 5.6: Progress of each beam during a TS based on candidate list.

5.7 Comparison of tabu search with and without using candidate list strategies

For comparison of the implemented TS algorithms, both are tested on 1000 randomly chosen beam setups. The results of this runs are shown in Figure 5.7. The black line within the shaded area is the 5 % trimmed mean of the objective function. The shaded area spans the distance from the first to the third quartile and the grey outer lines mark the minimum and the maximum values of the objective function.

Similar to ILS, using candidate list strategies and performing purposeful swaps results in a faster convergence for Equation (4.6). However, the discussed effect of a late increase of the improvement rate of ILS can not be observed for TS using candidate list solutions, since the algorithms initially improves beams by exchange of lamellas with other beams.

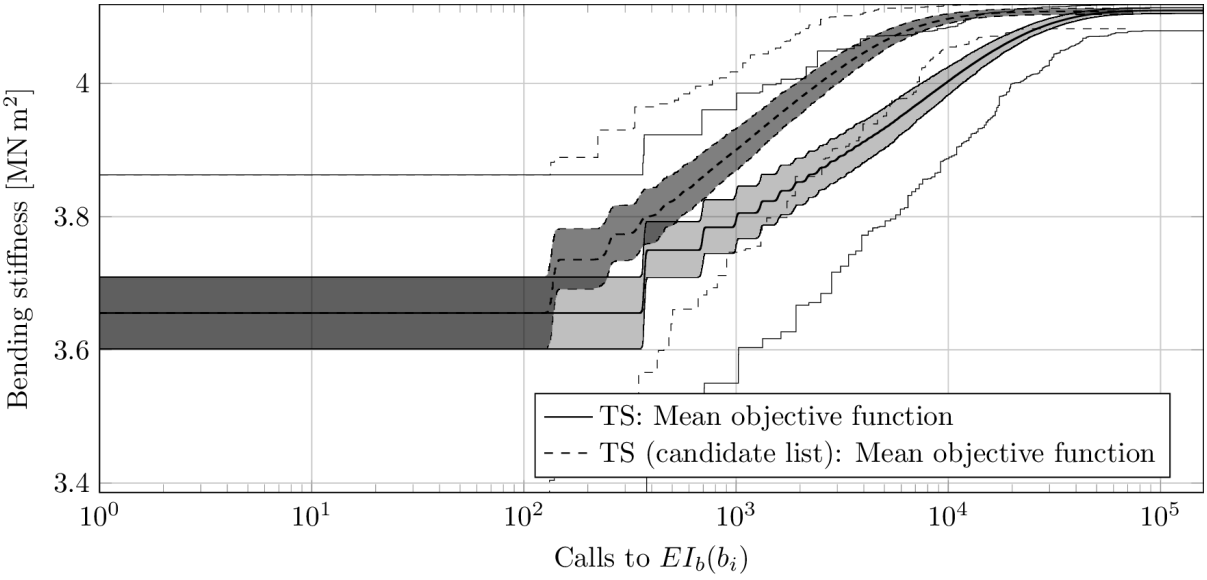


Fig. 5.7: Comparison of 1000 TS runs with and without the usage of candidate list solutions.

5.8 Application of genetic algorithms

The large number of different parameters and different operations applicable within GAs makes finding an optimal parameter set a challenging task. De Jong [10] discusses this topic on a general level and compares the usage of static and dynamic parameter setting strategies.

As the fitness landscape of the stated problem does not change during the run, online parameter tuning will not be used. De Jong [10] further proposes so-called parameter sweeps, where every possible value and combination of parameters is tested, in order to find an optimal static parameter setting. The computational effort for these tests for a number of different initial populations is, similar to the problem discussed in this thesis, very large. Therefore, the tuning process is done by using racing algorithms [35].

The basic idea of racing algorithms is, that every set of parameters (a model) is tested in parallel on different initial populations (a point) resulting in a value for the objective function (this can be seen as the error as usually optimisation is performed in form of a minimisation, thus the better the result the smaller the error). With every point tested on a model the average error of all tested points for this model is evaluated, thus giving an estimate of the models true error. By using statistical bounds the accuracy of this estimation is evaluated. In case the accuracy exceeds a certain threshold, models which are significantly worse than the best ones are discarded and the algorithm continues its focus on promising models. By employing this technique, it is not necessary to evaluate every possible parameter value and combination on a number of different starting populations as unpromising combinations are not tested in their entirety.

For this purpose, the free software package irace [32] is used. The parameters are bound to the following ranges and values[2]:

Population size:	20–100
Fitness scaling:	1.0–20.0; in case the selection function is SUS or roulette-selection
Mutation rate:	0.0–1.0
Crossover rate:	0.0–1.0
Elitism:	0–10
Crossover function:	PMX, OX, CX
Mutation function:	swap two lamellas, replace with random beam setup
Selection function:	Roulette-selection, SUS, tournament-selection, linear ranking
Tournament size:	2–10; in case the selection function is tournament-selection
Selection pressure:	1.0–2.0; in case the selection function is linear ranking
Chromosome type:	A or B
Initial population:	pure random or pseudo random

The parameters are tuned on the example from Section 5.1 on a number of 250 and 1000 generations. The maximum number of experiments is set to 10000, where one experiment is one optimisation run, with a given parameter model.

The 5 best parameter configurations are shown in Table 5.1. Herein the best parameter configurations are those which delivered, on average, the best results while maintaining a small inner quartile range, for both 250 and 1000 generations. The solution quality for the 5 parameter combinations is shown in Figure 5.8.

[2]Ranges delimited by integer numbers refer to integer numbers within the given ranges, whereas ranges not delimited by integers refer to all numbers within the given range with a precision of 2 decimal places.

The results clearly state which functions should be used to perform selection, crossover and mutation. Further, it is obvious that the pseudo random initial population and the chromosome type B outperform their counterparts.

Tab. 5.1: 5 best parameter combinations for the simplified problem

Parameter combination:	1	2	3	4	5
Population size N_p	90	100	90	90	87
Mutation rate p_m	0.96	0.88	0.92	0.89	0.93
Crossover rate p_c	0.85	0.79	0.95	0.84	0.98
Elitism ϵ_c	3	1	2	2	2
Crossover function f_c	Partially mapped crossover				
Mutation function f_m	Swap				
Selection function f_s	Tournament selection				
Chromosome type	B				
Initial population	Pseudo random initial population				
Tournament size t	8	6	7	7	7

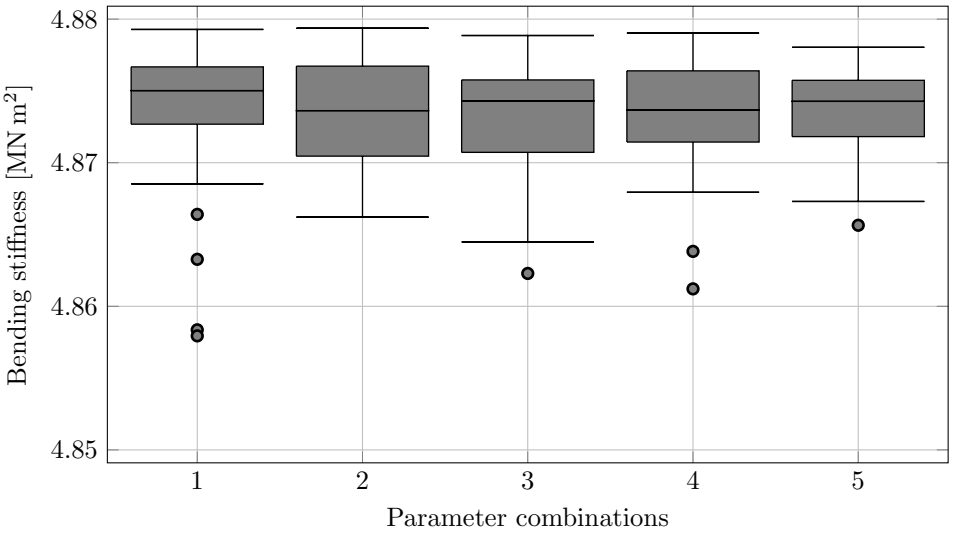


Fig. 5.8: Solution quality for the parameter combinations shown in Table 5.1

Figure 5.9 exemplary shows the optimisation procedure of the 5 different parameter combinations from Table 5.1. The computation time needed for the 5 different GA runs varies between 33 s and 47 s.

Figure 5.10 shows a comparison of the different parameter combinations starting from 100 different initial populations. Figure 5.10a shows the 5 % trimmed mean valued of the fitness function for the 5 best parameter configurations. As all 5 parameter configurations deviate only marginally from each other, the trend of the optimisation procedure in Figure 5.10b, is only shown for parameter configuration 1.

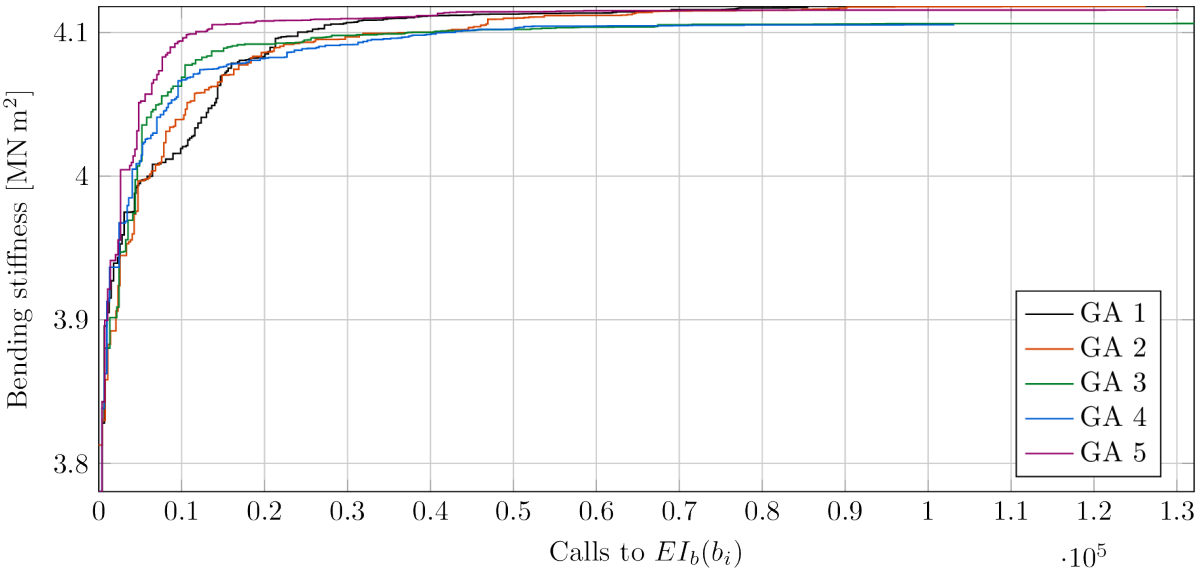
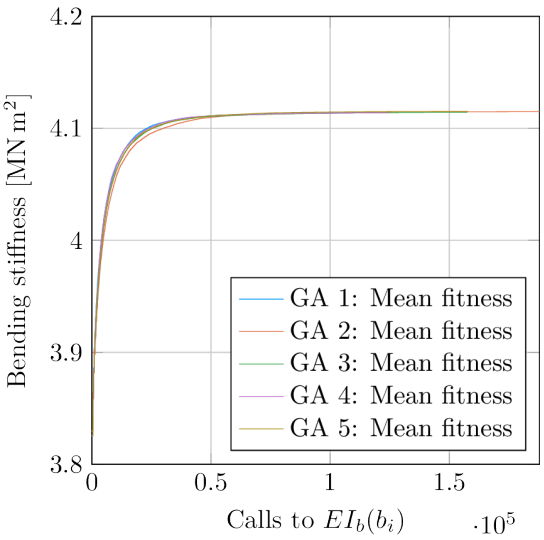
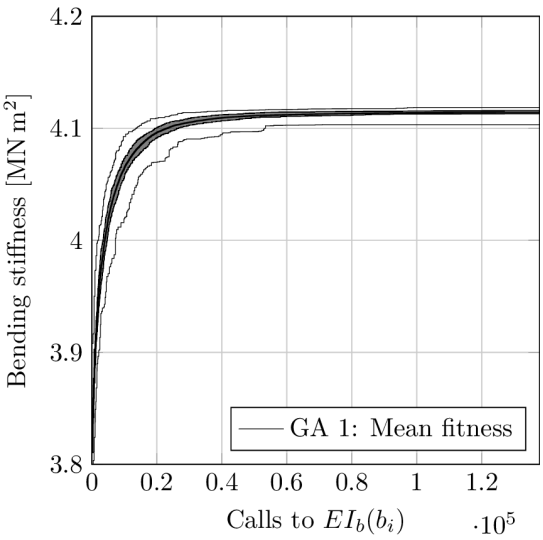


Fig. 5.9: Exemplary optimisation progress of the 5 best parameter combinations from Table 5.1



(a) Comparison of the 5 % trimmed mean value of the fitness functions for the 5 best parameter combinations



(b) Optimisation progress of 100 run for parameter combination 1

Fig. 5.10: Comparison of 100 optimisation runs using the 5 best parameter combinations

5.9 Comparison of algorithms for the simplified problem

Table 5.2 shows a summary of the results obtained from the MAs tested for the simplified problem. The 5 % trimmed mean of all discussed algorithms is shown in Figure 5.11. Based on theses results, it is possible to draw the following conclusions:

1. The best solution for Equation (4.6) is reached by LS. The other algorithms manage to reach similar values.

2. The worst solution is also given by LS, which roots in the discussed drawback of LS being trapped at local optima. This is further reflected in the standard deviation as LS has the highest value of all discussed algorithms.

   ILS and TS using candidate list solutions, in comparison to their counterparts, managed to escape from local optima, which is reflected in higher worst values and a slightly smaller standard deviation.

   In regards of the worst solution, GAs performed best, as the worst solutions are close to the average solutions of the other algorithms. This is further reflected in the smallest standard deviation of all discussed algorithms.

3. On average, GAs performed best, while ILS and TS using candidate list solution performed poorest.

4. ILS reached the smallest average value first. Both ILS and TS using candidate list solution converged significantly faster than their counterparts.

5. LS and ILS converged significantly faster than TS and GAs and still yield high performing solutions for the simplified problem. Regarding the average step count to reach the smallest value, GAs required 9.36 – 12.47 times more steps as ILS.

6. The exploratory capabilities of GAs outperform the ones of the others, as only a small portion of the runs never reach the smallest average value.

Generally all algorithms manage to find a near optimal solution to Equation (4.5) and all procedures prove the be applicable to the task of optimising GLT beams, since the relative error between the worst and the best solution is only 2 %. As for the simplified problem the complexity is reduced, it could be the case that exploitation is more important than exploration, thus LS still yields good solutions. Nevertheless, the effects of trying to maintain diversity within the solution space are reflected in a higher robustness of GAs and TS.

Tab. 5.2: Solutions of 1000 runs for each optimisation procedure. For the GAs the objective function was $f(\pi)$, whereas for the other approaches, the objective function reads $1/f(\pi)$, compare Equation (4.6).

	LS	ILS	TS	TS (cand.)	GA 1	GA 2	GA 3	GA 4	GA 5
Best[1]	**4.120**	4.118	4.119	4.119	4.119	4.119	4.118	4.119	4.119
Worst[1]	**4.038**	4.082	4.079	4.082	4.103	4.105	4.101	4.101	4.103
Avg.[1]	4.111	**4.108**	4.109	**4.108**	4.114	4.115	4.114	4.114	4.115
Std.[1]	0.007	0.006	0.006	0.006	0.003	0.003	0.003	0.004	0.003
Stepcount for reaching the smallest avg. value of 4.108									
Earliest	2511	1128	19393	4165	17997	19377	14958	20586	15353
Avg.	7151	3881	47339	13765	38196	48395	40836	36336	40202
Never[2]	26.9	38.9	36.9	44.0	6.0	7.0	8.0	9.0	6.0

[1] [MN m^2] [2] [%]

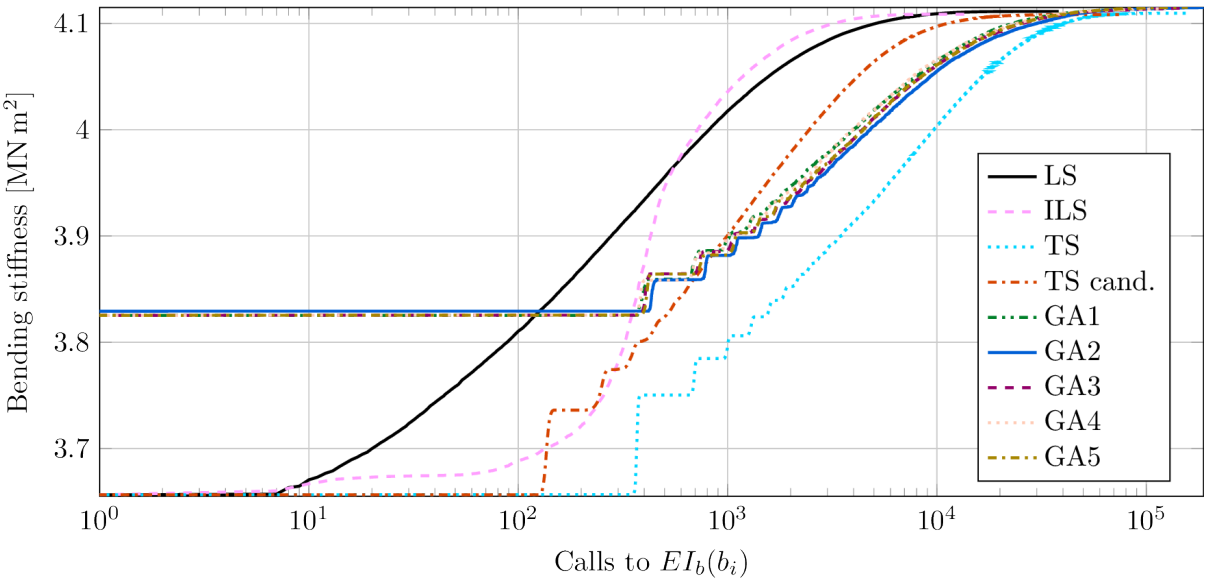


Fig. 5.11: Comparison of the 5 % trimmed mean values of the discussed metaheuristic algorithms

Chapter 6

Application of metaheuristic algorithms for the original problem

6.1 Finite element model

As stated in Section 1.2 and Section 4.2, for the actual problem it is necessary to consider that the lamella stiffness varies in longitudinal direction, which can be described by the stiffness profile $E(x)$. Therefore, it is not longer possible to perform the optimisation process based on Equation (4.5) but rather, as mentioned in Section 4.1.1, the deflection of the beam needs to be calculated by using a FE model.

The FE model used in this work is based on the one described by Kandler et al. [28]. The FE grid is constructed from 2D plane-stress elements based on quadratic shape functions. The element height is defined in such a way that each lamella consists of two elements in height, the greatest element length value is 25 mm. The bond between the layers is assumed to be perfect.

As stated by Kandler et al. [28], a transversal isotropic material behaviour is assumed. Further, only changes in the longitudinal stiffness profile $E(x)$ are considered, the Poisson's ratios ν_{21} and ν_{23} and the shear modulus G_{12} are assumed to be constant for every lamella. Thus, the stiffness tensor for each lamella reads

$$\mathbf{C} = \begin{bmatrix} \frac{E_1^2}{-E_2 \cdot \nu_{21}^2 + E_1} & \frac{E_1 \cdot E_2 \cdot \nu_{21}}{-E_2 \cdot \nu_{21}^2 + E_1} & 0 \\ \frac{E_1 \cdot E_2 \cdot \nu_{21}}{-E_2 \cdot \nu_{21}^2 + E_1} & \frac{E_1 \cdot E_2}{-E_2 \cdot \nu_{21}^2 + E_1} & 0 \\ 0 & 0 & G_{12} \end{bmatrix} \tag{6.1}$$

The values are obtained from the micro-mechanical model described by Hofstetter et al. [24], which is also employed in Kandler et al. [28]. The main parameters for the micro-mechanical model are mass density and moisture content. The mass density is known for each lamella and, at the time of evaluation, the moisture content was 12 %.

As the evaluation of the objective function, particularly the calculation of the deflection using the FE model, is crucial for the computation time of the optimisation process, the FE model is optimised as follows:

1. All finite elements have equal dimension.

2. One finite elements corresponds to exactly one lamella, i.e. it does not overlap or cross lamella boundaries.

3. Considering 1. and 2., it is possible to evaluate the element stiffness matrices for every finite element within every lamella once, before the actual optimisation algorithm is run. The global stiffness matrix can be assembled by matching the mesh coordinates with the lamella permutation L_i^* from Equation (4.2) or Equation (4.3).

4. Previously calculated solutions for the deflection $q_{\max}$, mentioned in Section 4.1.1, are stored. In case some parameter set is used at a later stage, instead of a time-expensive recomputation, the algorithm can access the result from memory. This is especially useful as the described MAs mostly perform small changes or tend to exploit already well-known search areas in later stages of the optimisation process. This can easily be illustrated by examining the neighbourhood definition Γ for LS, introduced in Section 5.2:

   Γ consists of beam setups generated by performing every possible swap of two lamellas. For the stated example from Section 5.1, Γ consists of $\frac{(n_l-1)\cdot n_l}{2} = 1225$ solutions, where each solution contains 5 beams. Therefore, the neighbourhood consists of a total of 6125 beams, which have to be analysed. Assuming that the beams contained in the current solution s have all been calculated, a swap can generate one new beam, in case two lamellas within a beam were swapped, or two new beams, in case two lamellas between two beams were swapped. The number of swaps resulting in one newly generated beam are, similar to the total number of solutions, given by an arithmetic series. For the practical example this results in $\frac{(n_{b,l}-1)\cdot n_{b,l}}{2} \cdot n_b = 225$ solutions which result in one new beam. Hence, the remaining 1000 solutions generate two new beams each, therefore Γ consists – in the worst case – of 2225 previously unknown beams which is approximately only a third of the beams contained in Γ.

5. The outer distances a_3 shown in Figure 5.1 are not taken into account as they have no impact on the load-bearing behaviour of the beam.

Figure 6.1 shows the resulting stresses for a completely random beam. The effect of the varying longitudinal stiffness profile $E(x)$ is clearly visible as darker areas in the beam, corresponding to weaker sections, attract less stresses. Consequently the strains in those areas are greater than in the surrounding region.

6.1.1 Benchmark tests

Due to the high impact of the objective functions computation time on the computation time needed for the overall optimisation progress, a benchmark test of the described FE model is performed. In combination with the number of needed objective function evaluations from Section 5.1, a first estimate of the time needed for the optimisation can be made.

The evaluation of the objective function involves the following steps:.

1. Assembly of the mesh based on the lamella permutation L_i^*,

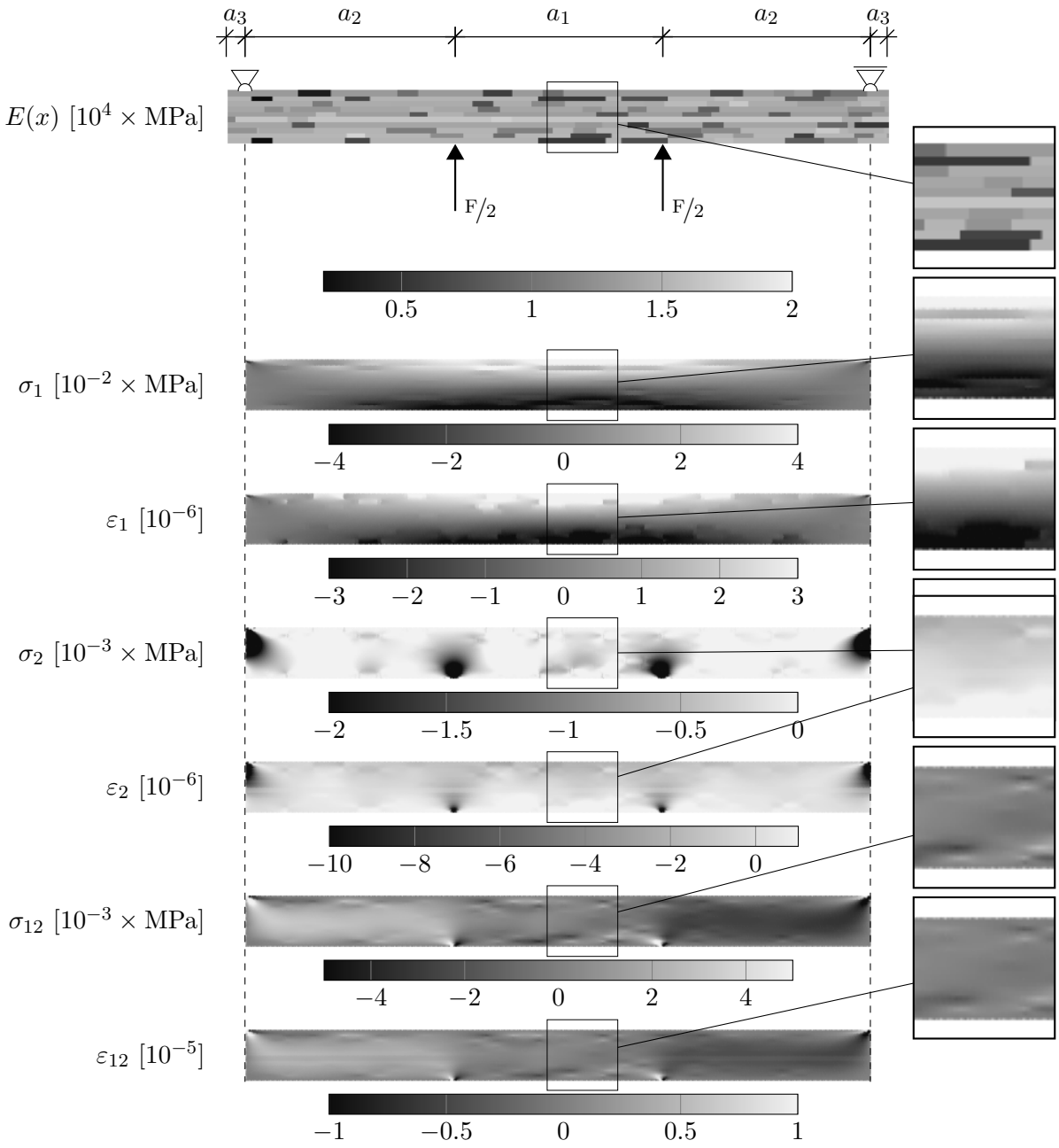


Fig. 6.1: FE model results of a completely random GLT beam

2. assembly of the global stiffness matrix and
3. solving the system equilibrium $\mathbf{K} \cdot \vec{q} = \vec{p}$.

To eliminate random effects due to background processes, the times needed for each step are recorded for 1000 different beams. The results of those tests are displayed in Table 6.1. All computations were made on a 2015 MacBook Pro with a 3.1 GHz Dual-Core Intel i7 CPU and 16 GB RAM.

Referring to the number of distinct objective function evaluations from Section 5.1, ranging from 1.2×10^4 for ILS to 1.3×10^5 for GAs, the total computation time ranges from 59 minutes to 633 minutes.

Tab. 6.1: Resulting average and standard deviation of the computation time, as well as its relative amount of the benchmark tests of 1000 different beams. Step 1 refers to the mesh assembly, step 2 to the assembly of the global stiffness matrix and step 3 to solving the system equilibrium equation.

	Avg. time [ms]	Share on time [%]	Std.dev. of time [ms]
Step 1	0.8	0.3	0.3
Step 2	80.9	27.7	56.4
Step 3	210.6	72.0	38.1
Total	292.3	-	72.8

As this approach only considers the computation time of the mechanical model and thus, is certainly the lower bound of execution time, an alternative method for approximating the results from the FE model is needed.

6.2 Approximation of the Finite element model

As shown in Section 6.1.1, the computational effort for evaluating the FE model is too large to be used efficiently with the proposed MAs. Coelho et al. [7] suggests approximating the FE model with an online learning regression model further referred to as metamodel. The term online learning refers to the task of continuously updating the metamodel, instead of defining it once. Coelho et al. [7] emphasize the usage of online learning, since during the optimisation process the solutions space is narrowed down to a set of solutions close to the optimum. Thus, the learning algorithm should change its focus to this new part of the solution space. Details on the implementation and usefulness of online learning are given in Section 6.2.4.2 and Section 6.2.5.2.

In the following sections, two different types of learning algorithms are described and adapted to the given problem. Section 6.2.4 describes the implementation of a so-called eager learner based on a least squares problem [27, p. 245]. Section 6.2.5 describes the implementation of a so-called lazy learner using k-nearest-neighbour (kNN) [9] classification.

The main difference between eager- and lazy learning algorithms is in the time of evaluation of the training data. Aha [1] characterises the main differences between lazy and eager learning algorithms as follows:

Lazy learners initially only store the input. The process of making a prediction is performed at time a request is made. Eager learners, on the other hand, build the metamodel beforehand, based on the complete available information set and defer the input data after the training process is finished. An advantage of eager learners is that, since the computationally expensive learning phase is performed beforehand, predictions can be made – compared to lazy learners – very fast. Nevertheless, the lazy learner can profit from postponing the generation of the metamodel, as every model is based on the characteristics of the vicinity of the query point, which can significantly improve the prediction quality.

6.2.1 Training- and test set

Both types of algorithms performed training on 5000 beams generated based on the described example from Section 5.1. For validation, a test set of another 500 beams is generated.

Both of the sets are created by the pseudo random initial population generator mentioned in Section 4.8.6 to achieve well distributed lamella permutations.

6.2.2 Comparing beams

The lazy learner as well as the eager learner need a beam representation on basis of which the algorithms are capable of comparing beams or performing calculations. For the purpose of comparing beams, it is possible to calculate the similarity between two lamella permutations L_a^* and L_b^* by comparing their integer representations mentioned in Section 4.3. This method solely considers lamella numbers and orientations and completely neglects the similarity between lamellas. E.g in case two lamellas have identical material parameters this method is not capable of recognizing their similarity.

An alternative method for comparing beams or lamellas can be implemented by comparing the corresponding stiffness profile $E(x)$. This is achieved by sampling the $E(x)$ profiles with a given resolution and use the result as a row vector. A beam can be described by a matrix $\mathbf{B}_i$ build up from the row vectors representing the corresponding lamellas as follows:

$$\mathbf{B}_i = \begin{bmatrix} E_{i,1,1} & E_{i,1,2} & \dots & E_{i,1,N_x} \\ E_{i,2,1} & E_{i,2,2} & \dots & E_{i,2,N_x} \\ \vdots & \vdots & \dots & \vdots \\ E_{i,n_{b,l},1} & E_{i,n_{b,l},2} & \dots & E_{i,n_{b,l},N_x} \end{bmatrix}, \tag{6.2}$$

where $E_{i,j,k}$ is the value of the sampled $E(x)$ function for the lamella at position j in vector L_i^*. N_x denotes the number of samples generated based on the resolution. The value for $E_{i,j,k}$ is calculated by computing the moving average with a window size of $\frac{l_b}{N_x-1}$ where l_b denotes the beam length. Respectively for $E_{i,j,1}$ and E_{i,j,N_x} the window size is $\frac{l_b}{2\cdot(N_x-1)}$. Alternatively, a nearest neighbour interpolation could be implemented. However, for the present case, a moving average is more suitable since, in order to reduce the computational effort, N_x is chosen to be small opposed to the number of FE used for one lamella. Thus, in case a knot area is between

two sample points, such that the nearest neighbour interpolation is not detecting it, the knot vanishes.

For being able to perform operations in a vector space, for $\mathbf{B}_i$ the alternative vectorial representation is chosen:

$$\vec{B}_i = \begin{pmatrix} E_{i,1,1} \\ E_{i,1,2} \\ \vdots \\ E_{i,1,N_x} \\ E_{i,2,1} \\ E_{i,2,2} \\ \vdots \\ E_{i,2,N_x} \\ \vdots \\ E_{i,n_{b,l},1} \\ E_{i,n_{b,l},2} \\ \vdots \\ E_{i,n_{b,l},N_x} \end{pmatrix}. \tag{6.3}$$

6.2.3 Parameter sensitivity and weighting

In Section 4.1.2.1 and Section 4.2, it was shown that by using strong lamellas in the outer layers of a beam, the resulting beam bending stiffness is increased. By implication this means, outer layers are of more importance for the resulting bending stiffness or deflection of the beam. Thus, to be able to compare beams in the context of the deflection $q_{\max}$, resulting from the FE model, the effect of knots at different locations within the beam are examined. The corresponding results will also provide the eager learner with an initial solution for the weighting vector. The reasoning behind weighting the $E_{i,j,k}$ values in $\vec{B}_i$ for the lazy learner, is to reduce $\vec{B}_i$ to the characteristics relevant for the resulting deflection $q_{\max}$ and thus providing a proper basis for comparing different beams. The geometry and loads for the FE model are used from the example in Section 5.1.

The resulting position dependent sensitivity is stored in a matrix similar to $\mathbf{B}_i$ in Equation (6.2). This sensitivity matrix is obtained by extracting two different element stiffness matrices $\mathbf{KE}_{cw}$ from the clearwood section and $\mathbf{KE}_{knot}$ from a random knot section. Both matrices are extracted from lamella 59 of the grading class LS22. Subsequently 100 random variations of $\mathbf{KE}_{knot}$ are generated from:

$$\mathbf{KE}_{i,knot} = \mathbf{KE}_{knot} - 0.5 \cdot \mathbf{KE}_{knot} \cdot X, \tag{6.4}$$

where X is a random variable uniformly distributed between 0 and 1. It should be noted that this relation yields different results as if the modulus of elasticity in longitudinal direction, E_1, was varied. However, in a realistic scenario it is reasonable to assume also variations in the other

material parameters E_2, G_{12} and ν_{21}. Also, this approach is only used to assess the sensitivity, thus the approximation error introduced by Equation (6.4) can be accepted.

To gain insight on how a weak zone i at a defined location (x, y) affects the resulting maximum deflection, the element stiffness matrix $\mathbf{KE}_{i,knot}$ is assigned to the corresponding finite element at location (x, y) and – to guarantee a reasonable weak zone size – to two horizontal neighbours within the beam mesh. The element stiffness matrix $\mathbf{KE}_{cw}$ is assigned to the remaining elements. For the resulting system, the maximum deflection is computed and stored for the given location (x, y) and the weak zone i. This step is repeated for all 100 random variations of $\mathbf{KE}_{knot}$ for all y at 8 pre-defined vertical axis uniformly distributed between the left end and the centre of the beam (utilizing the symmetry of geometry and loads). The values between the 8 vertical axis are linearly interpolated.

A measure of sensitivity of $q_{\max}$ to a certain location is obtained through the standard deviation of all $q_{\max}$ resulting from all 100 $\mathbf{KE}_{i,knot}$ at this location. The underlying idea is that the GLT system is insensitive to knot sections at locations with a low standard deviation of $q_{\max}$. Consequently, the system is sensitive to knot areas at locations where a high standard deviation of $q_{\max}$ is observed.

To be able to to use the obtained weighting independently of $\mathbf{KE}_{i,knot}$ and $\mathbf{KE}_{cw}$, the weighting vector is normalized by the maximum standard deviation. Furthermore, to apply the vector to different geometries the width is scaled to a range from 0 to 1 and the height to a range from -1 to 1. It should be noted, that in case of a change in the boundary conditions, e.g. supports at different coordinates, the sensitivity analysis has to be recomputed.

Figure 6.2 shows the resulting normalized weighting function for the given problem. It is clearly visible that locations further from the centre of mass of the beam and closer to the middle of the beam have a higher impact on the deflection.

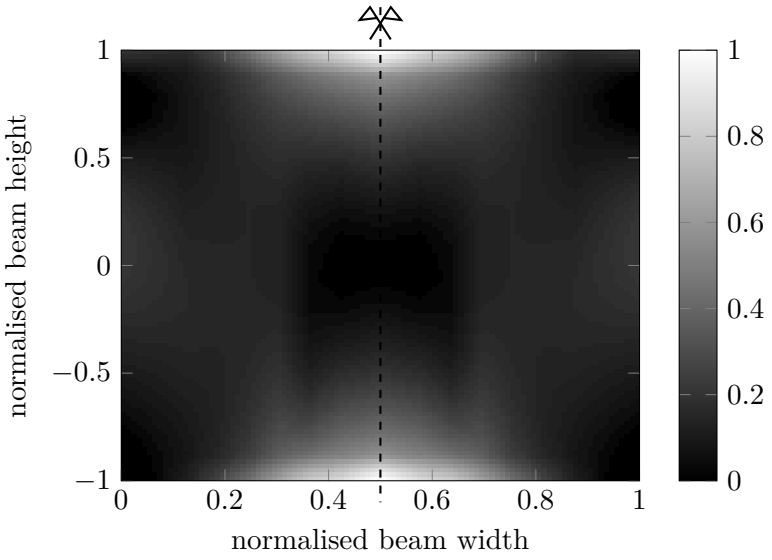


Fig. 6.2: Position dependent normalised weights for variations in $E(x)$ affecting $q_{\max}$

Similar to the matrix representation for a beam derived in Section 6.2.2 the obtained weights can be sampled into a weighting matrix $\mathbf{W}$. For being able to perform operations in a vector space, $\mathbf{W}$ is stored in a vectorial representation $\vec{W}$, based on the same logic as used in Equation (6.3).

6.2.4 Eager learner

The basic idea behind the implementation of the eager learner is that the FE model calculations can be approximated solely based on the vector $\vec{B}_i$. The resulting approximate deflection $q_{\max}$ is calculated by performing an inner product on the vector $\vec{B}_i$ and a weighting vector $\vec{w}$:

$$q_{\max,\text{approx.}} = \vec{B}_i \cdot \vec{w} \tag{6.5}$$

The training is performed based on two different variations of $\vec{B}_i$.

1. Using the inverse of every entry in $\vec{B}_i$. This is insofar arguable as a higher modulus of elasticity reduces the deflection. Thus, the index originally representing the k-th sampled element, in the lamella at location j in beam i is used as follows:

$$\vec{B}_i[j,k] = \frac{1}{E_{i,j,k}} \tag{6.6}$$

2. In addition to the variation in 1., weighting every element $E_{i,j,k}$ by $\frac{E_{i,j,k}}{E[\vec{B}_i]}$ where $E[\vec{B}_i]$ is the average of all values in $\vec{B}_i$. This weighting is indented to simulate the effect that stiffer regions tend to attract stresses. Thus, the index originally representing the k-th sampled element, in the lamella at location j in beam i is used as follows:

$$\vec{B}_i[j,k] = \frac{1}{E_{i,j,k}} \cdot \frac{E[\vec{B}_i]}{E_{i,j,k}} \tag{6.7}$$

To obtain the unknown values of the weights $\vec{w}$, a least squares approach [27, p. 245] is used:

$$\text{Minimize: } L = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{2}\left(q_{\max,i} - \vec{B}_i \cdot \vec{w}\right)^2 + \underbrace{\frac{\alpha}{2}\sum_{j=1}^{N_x \cdot n_{b,l}}\left([w_j]^-\right)^2}_{\text{Constraint } \forall w_j | w_j \geq 0}\right), \tag{6.8}$$

where n is the number of beams in the training set, w_j is the j-th element in the weighting vector $\vec{w}$, α is the scaling factor for the non-smooth penalty function and the notation $[\bullet]^-$ is defined as $\max\{0, -\bullet\}$ [27, p. 507]. The non-smooth penalty function constrains w_j to positive values. The constraint for only positive values of w_j, assures that $\vec{B}_i \cdot \vec{w}$ stays positive, i.e. the deflection returned by the learner cannot be negative. Nevertheless, the eager learner is also tested for $\alpha = 0$.

The gradient of L for w_j results in:

$$\frac{\partial L}{\partial w_j} = \begin{cases} \frac{1}{n}\sum_{i=1}^{n}\left(\left(q_{\max,i} - \vec{B}_i \cdot \vec{w}\right) \cdot (-B_{i,j})\right) & \text{if } w_j > 0 \\ \frac{1}{n}\sum_{i=1}^{n}\left(\left(q_{\max,i} - \vec{B}_i \cdot \vec{w}\right) \cdot (-B_{i,j}) + \alpha \cdot w_j\right) & \text{if } w_j < 0 \end{cases} \tag{6.9}$$

where $B_{i,j}$ is the j-th element in vector $\vec{B}_i$.

The actual values for $\vec{w}$ are obtained by performing a numerical optimisation by using the Limited-memory-BFGS (L-BFGS) algorithm [27, p. 177]. L-BFGS is based on the BFGS algorithm which is a quasi-Newton method. Quasi-Newton methods usually approximate the hessian matrix, needed for estimating the next step of the optimisation. More precisely, in case of the BFGS algorithm, the inverse of the hessian matrix is approximated.

Since the hessian matrix contains all second order derivatives for all variables subject to the optimisation, storage and manipulation is usually very memory intensive. Thus, L-BFGS stores the inverse of the hessian implicitly, by storing a certain number of vectors its constructed from (usually between 3 and 20 [27, p. 177]). Though L-BFGS approximates the second order derivatives, the gradient $\frac{\partial L}{\partial w_j}$ is reguired.

The gradient function used for L-BFGS is computed by using stochastic gradient descent (SGD) [6] on a random number of beams from the training set. The usage of SGD reduces the number of calls to the gradient function by calculating the gradient based on a portion of the training set. Furthermore, the added stochastic effect reduces the chance that the optimisation gets stuck at a local minimum.

6.2.4.1 Results

As stated above, the eager learner is trained based on the two different variations of $\vec{B}_i$. Additionally, for each variation the constraint α is either respected or ignored (i.e. $\alpha \neq 0$, $\alpha = 0$). Therefore, in total, four different parameter combinations, as shown in Table 6.2, are tested.

Tab. 6.2: Parameter combinations for the eager learner

	SGD size[1]	α	w_0[2]	Var. of $\vec{B}_i$	Resolution[3]
1	50	0.0	10^{-3}	1	50
2	50	0.0	10^{-3}	2	50
3	50	1.0	10^{-12}	1	50
4	50	1.0	10^{-12}	2	50

[1] The number of samples for which the gradient is calculated. [2] w_0 is initially populated based on the sensitivity vector from Section 6.2.3, scaled by the given factor. [3] The resolution is given in number of segments in longitudinal direction per lamella.

The performance of the learning algorithm is tested with a test set of 500 beams and compared against the deflection obtained by the FE model. The quality of the approximation is determined

by the so-called coefficient of determination [11, p. 484] denoted by R^2. The coefficient of determination measures the precision of a linear regression model of explaining variation in the input data. For $R^2 = 1.0$ the linear regression model perfectly fits the sample data, for $R^2 = 0.0$ the model cannot explain any variations in the sample data. As can be seen from the results in Figure 6.3 parameter sets 1 and 3 (using no further weighting of $E_{i,j,k}$), perform better on the test set.

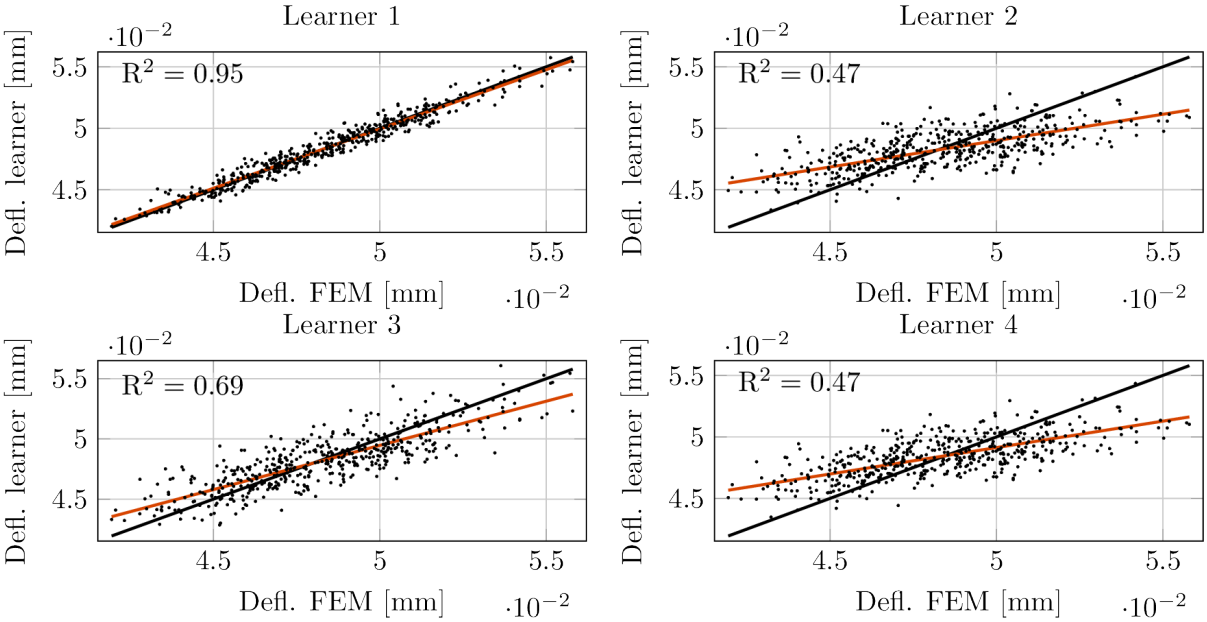


Fig. 6.3: Approximated deflection of the test set compared against the deflection obtained by the FE model

Especially for GAs the absolute value of the deflection is not crucial, as solutions are compared against each other. Rather, it is more important that the approximation is able to correctly predict whether one beam is better than another one, for a large number of beam pairs. Thus, the performance is further rated by comparing the results of a tournament selection, as described in Section 4.8.1, using a tournament size of $t = 6$. The selection process is repeated 20000 times to reduce the stochastic effects involved in the selection method. Figure 6.4 shows the results in form of a probability density function, indicating if a certain beam is a member of the newly generated population. Thicker lines illustrate the density based on the FE model calculation, thinner lines the approximations using the eager learner. The quality of the approximation is quantified by the correlation coefficient [11, p. 209] of the density functions, resulting from the FE model and the learning algorithm. The correlation coefficient ranges between -1 and 1, where -1 is a total negative linear correlation, 0 is no linear correlation and 1 is a total positive linear correlation.

Similar to Figure 6.3, Figure 6.4 shows that the parameter sets 1 and 3 outperform the others. Furthermore, even though the learning algorithm is entirely unfamiliar with the test set, the selection based on the approximate model matches the one based on the FE model well.

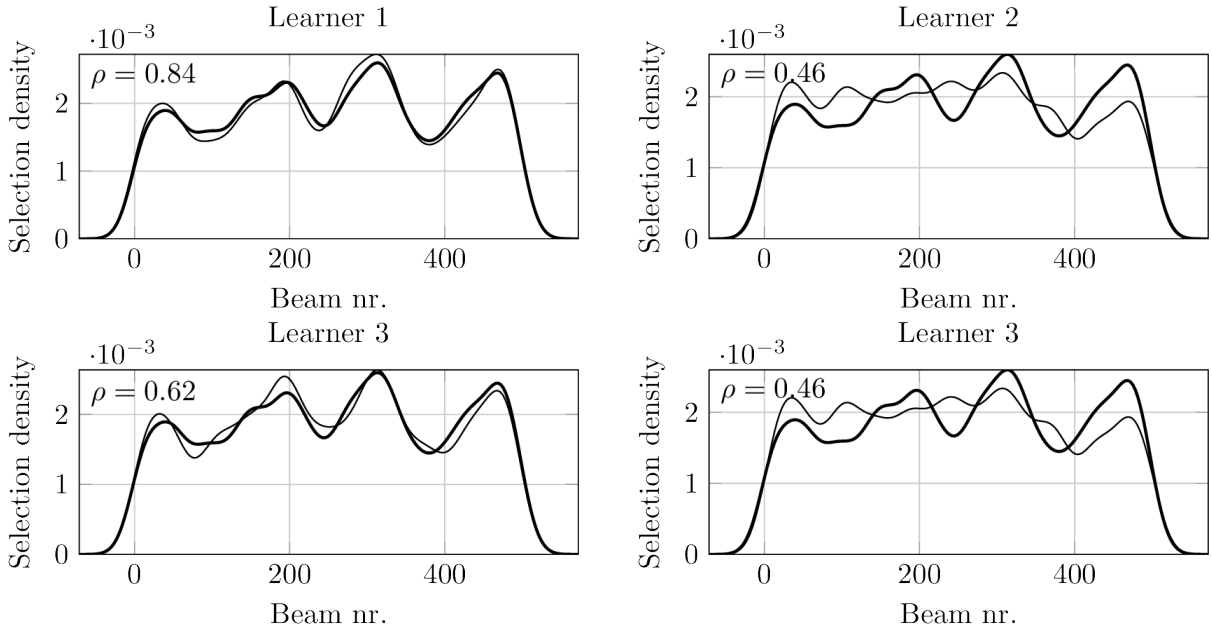


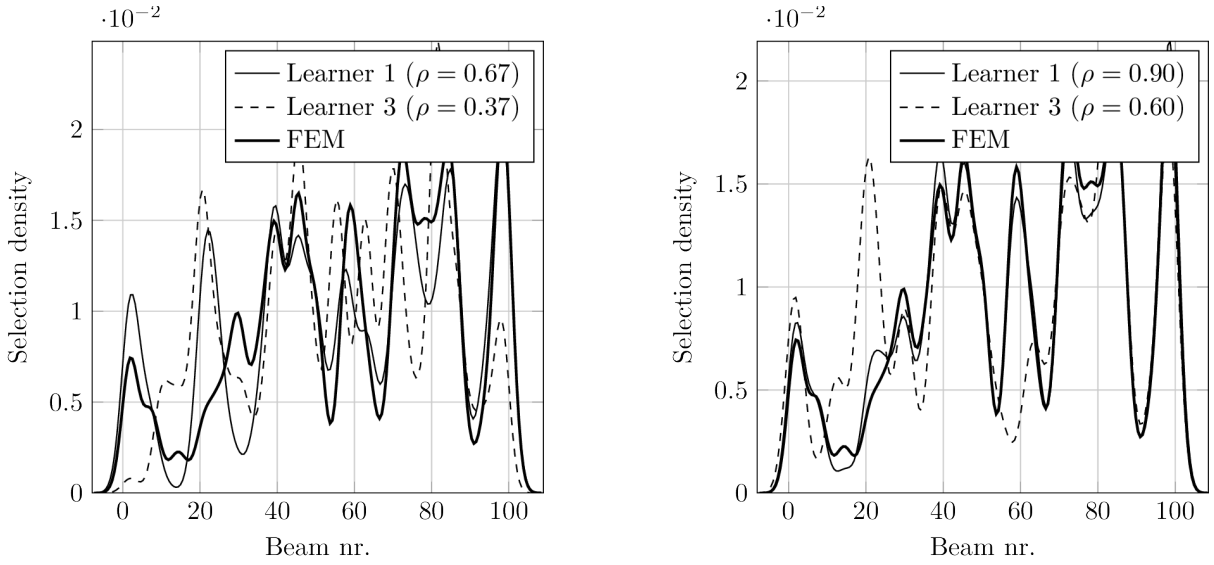
Fig. 6.4: Probability density of beams being part of the selected population when using an eager learner compared against using the deflection obtained by the FE model

6.2.4.2 Online learning

The above statement about the eager learners performance is true for the given random test set, which consists of well distributed lamella permutations. However, as already mentioned, in later runs of MAs, solutions tend to become more similar. Therefore, the learning algorithm is tested on another set of beams generated by selecting out of 10000 pseudo-randomly generated beams the 100 best beams, i.e. the ones with the lowest deflection. For the reduced test set, $q_{\max}$ has a coefficient of variation of 3.35×10^{-4}, whereas for the original test set of 10000 beams, $q_{\max}$ has a coefficient of variation of 553.82×10^{-4}. Once again tournament selection as described Section 6.2.4.1 is performed for both the learning based approximation and the FE model solution.

Figure 6.5a shows that the eager learner, both with and without positive-weight-constraint, is not longer capable of distinguishing the differences between the beams. Therefore, to sustain the performance of the learning algorithm, online learning is applied to sufficiently approximate the narrowed down solution space. A detailed description on how online learning is incorporated with the optimisation procedure, is given in Section 6.3.1.

Figure 6.5b shows parameter sets 1 and 2 after a retraining phase on the new training set. Parameter set 1 now outperforms parameter set 3 when comparing the results shown in Figure 6.5b. Another benefit of using parameter set 1 is that during the initial learning phase, set 1 converged after 1674 calls to the objective function whereas set 3 needed 7047 calls. This is manly due to the missing constraint for parameter set 1. The resulting negative weights in $\vec{w}$, seem to be compensated for the vectors $\vec{B}_i$ in the current context. Therefore, in the following steps, only parameter set 1 is used for the eager learner.



(a) Tournament selection before retraining

(b) Tournament selection after retraining

Fig. 6.5: Probability density of beams being part of the selected population when using an eager learner compared against using the deflection obtained by the FE model for beams having similar deflection

6.2.5 Lazy learner

kNN methods as described by Cover et al. [9] find application when trying to classify unclassified samples based of their resemblance to previously classified samples. This classification method infers the class of the unknown sample, based on the k samples with the most resemblance to the unknown sample. In this work, however, an adapted version is incorporated which is able to perform regression instead of classification tasks. In case of regression, each stored sample is assigned a numerical (usually real) value. For the stated problem, each beam is represented in form of a matrix $\mathbf{B}_i$ and $q_{\max}$ is obtained from the FE model. To predict the deflection $q_{\max}$ for a given beam without the need to employ the computationally expensive FE model, the regression is performed by inference from the k nearest neighbours with already known values. Herein, by k nearest neighbours it is referred to the k most similar specimens. The similarity between samples is usually expressed in the distance d_i, where d_1 is the distance to the sample closest to the unknown sample and d_k the distance to the furthest.

For larger values of k it is often useful to define a distance weighting function, as proposed by Dudani [12], to consider the effects of more similar samples and less similar samples within the group of the k selected samples. The method used herein introduces the weight w_i for the k values $q_{i,max}$ as follows:

$$w_i = \frac{1}{d_i},\ d_i \neq 0. \tag{6.10}$$

The resulting equation for the approximation $q'_{\max}$ can be formulated as

$$q'_{\max} = \frac{\sum_{i=1}^{k} w_i \cdot q_{i,max}}{\sum_{i=1}^{k} w_i}. \tag{6.11}$$

For finding the k most similar samples and calculating w_i, the "distance" between two beams $\vec{B}_i$ and $\vec{B}_j$ is calculated employing the vectorial beam representation introduced in Section 6.2.2. Using a weighted euclidean distance, the distance $d_{i,j}$ between the beams i and j is defined as

$$d_{i,j} = \sqrt{\sum_{k}^{N_x \cdot N_{b,l}} (B_{i,k} - B_{j,k})^2 \cdot W_k}, \tag{6.12}$$

where $B_{i,k}$ is the k-th feature of $\vec{B}_i$, $B_{j,k}$ is the k-th feature of $\vec{B}_j$, and W_k is the k-th feature of the weighting vector $\vec{W}$ from Section 6.2.3.

6.2.5.1 Results

Similar to the eager learner, training is performed on the training test set by adding the sample data to the knowledge of the lazy learner.

Since the performance of the kNN regression strongly depends on k, the learner is tested for values of 5, 15, 25 and 50 nearest neighbours.

Figure 6.6 shows the deflections obtained from the lazy learner compared against the deflections obtained from the FE model. The results for different values of k are very similar which might be due to the inverse distance weighting. With increasing values of k a slight drift from the FE-based deflection rotation of the scattered points and the linear regression curve is noticeable. This can be explained by k approaching the total number of stored samples and thus, the results obtained by the learner consist of information based on the same samples. Considering a case without inverse distance weighting, where the learners prediction is made by calculating the mean value of the k nearest neighbours for larger k every prediction would be equal.

Figure 6.7 shows the result from tournament selection, performed on the test set. Once again, the variations due to different values of k are minimal.

6.2.5.2 Online Learning

The reasons for using online learning for the eager learner are also applicable to the lazy learner. The tournament selection on the test set of 100 beams (as described in Section 6.2.4.2) results in the probability density shown in Figure 6.8. Similar to the eager learner, the original lazy learner performs poorly on similar beams. The online learning for the lazy learner is not implemented in form of a re-training phase but by adding the newly found solutions to the information storage of the lazy learner. Thus, the density functions for the retrained learner are not plotted, as the learner just returns the exact results of the FE model.

The computation time of a prediction from the lazy learner strongly depends on the number of samples contained in its storage, as for every prediction the distance between the unknown sample and the stored samples needs to be calculated. By removing samples which are hardly contained in solutions, the prediction time can be heavily reduced. This is implemented by keeping track on how often a sample occurred within a solution. When the ratio between this

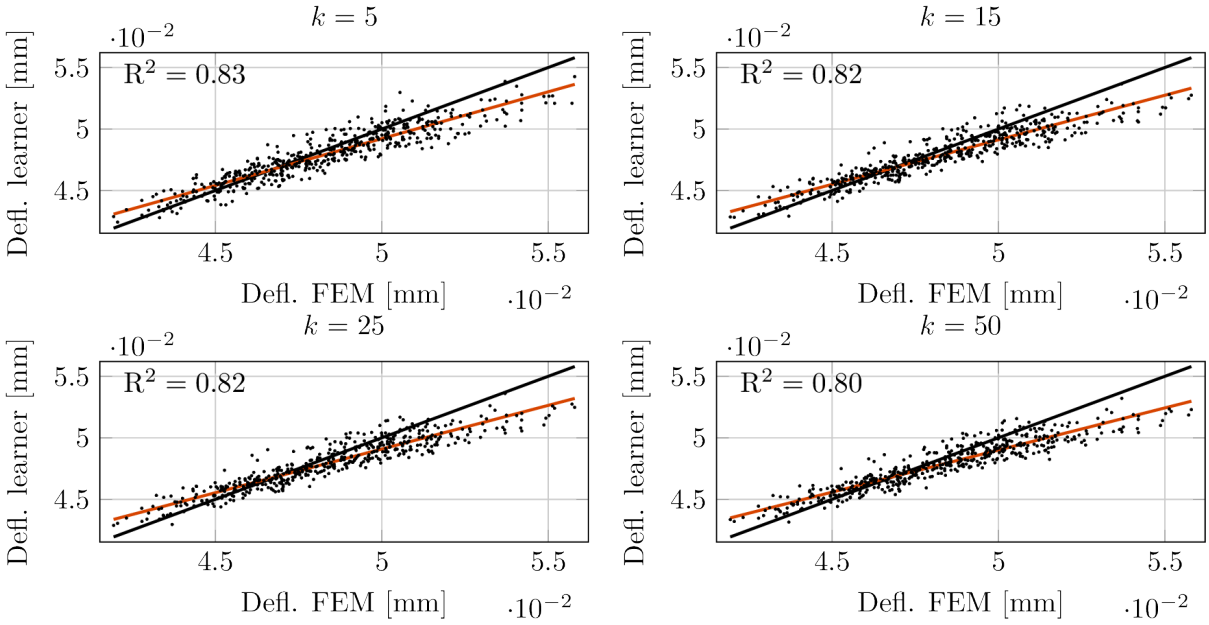


Fig. 6.6: Approximated deflection of the test set compared against the deflection obtained by the FE model

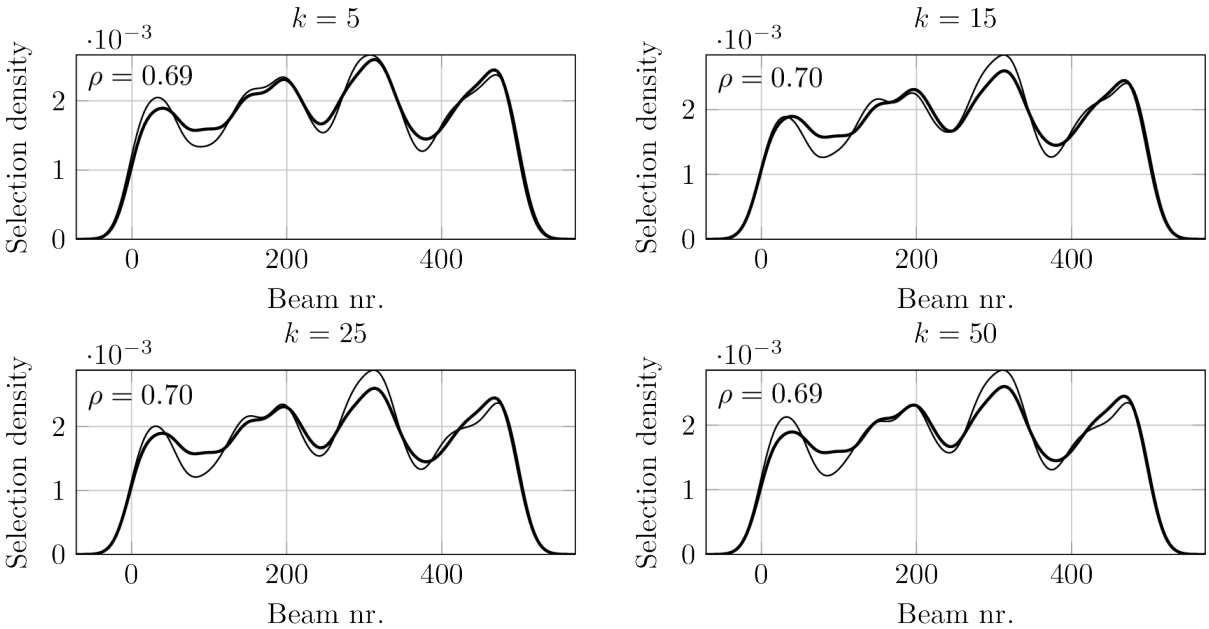


Fig. 6.7: Probability density of beams being part of the selected population when using a lazy learner compared against using the deflection obtained by the FE model

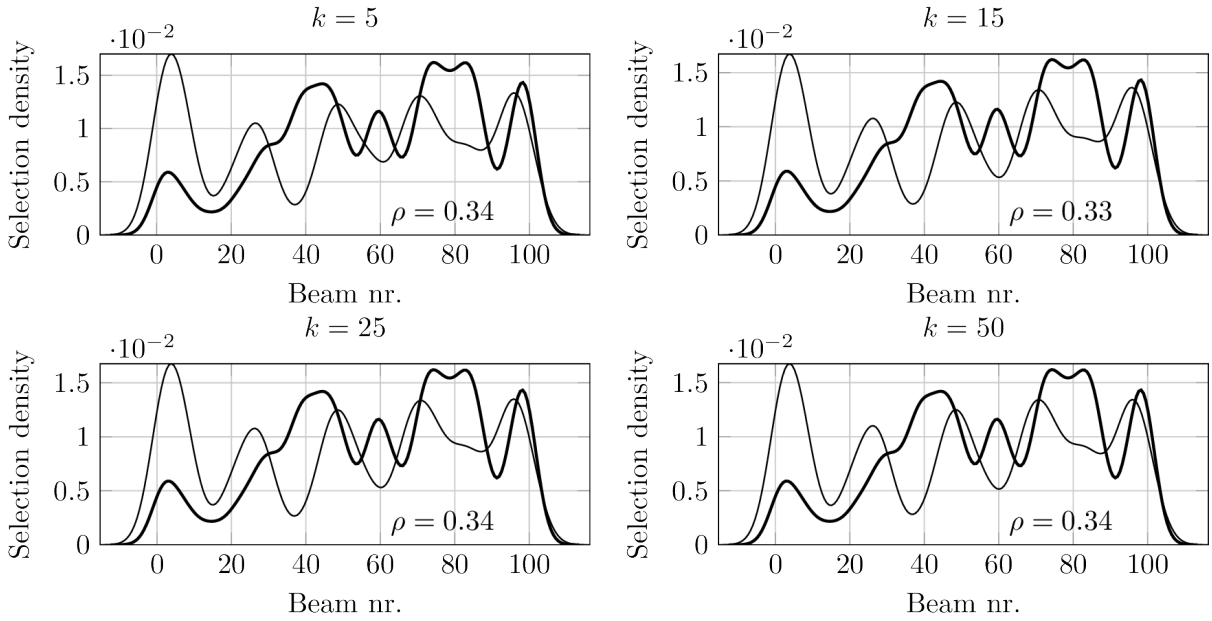


Fig. 6.8: Probability density of beams being part of the selected population when using a lazy learner, compared against using the deflection obtained by the FE model, for beams having similar deflection

number and the total number of predictions of the learner falls below a given threshold, the sample is discarded.

6.3 Implementation of metaheuristic algorithms for the non simplified problem

6.3.1 Implementation of online learning

Section 6.2.4.2 and Section 6.2.5.2 describe how online learning is performed on the learning algorithms level. Yet how and when it is performed on the optimisation algorithms level has not been defined. The implementation depends on the one hand on whether an eager or a lazy learner is used, and on the other hand if the type of optimisation algorithm is population based or not. Furthermore, it is important to notice that, by performing online learning, equal solutions can result in different values for the objective function at different stages of the optimisation progress. Hence after a retraining phase it is crucial to update all stored solutions from earlier stages which will be compared to solutions in later stages, e.g. the currently best known solution or solutions used for convergence criteria.

6.3.1.1 Eager learning for population based algorithms

The advantage of population based algorithms like GAs is that both the learning algorithm and the optimiser can operate on the same defined set of samples. Hence the learner is able to perform training on the solution space the optimiser is likely to explore.

This is implemented as follows: The eager learner starts pre-trained on the training samples described in Section 6.2.1. The optimisation is then performed until the convergence criterion is met. Afterwards retraining is performed on the last population and the optimisation process is continued. Subsequently this process is repeated until no further improvement is achieved.

6.3.1.2 Lazy learning for population based algorithms

The online learning for the lazy learning algorithm is implemented such that every K steps the optimisation process is interrupted and the deflection is calculated for N_r beams from the best performing populations, using the FE model. The solutions obtained by the FE model are then added to the knowledge of the lazy learner and the optimisation process is continued.

6.3.1.3 Learning for non population based algorithms

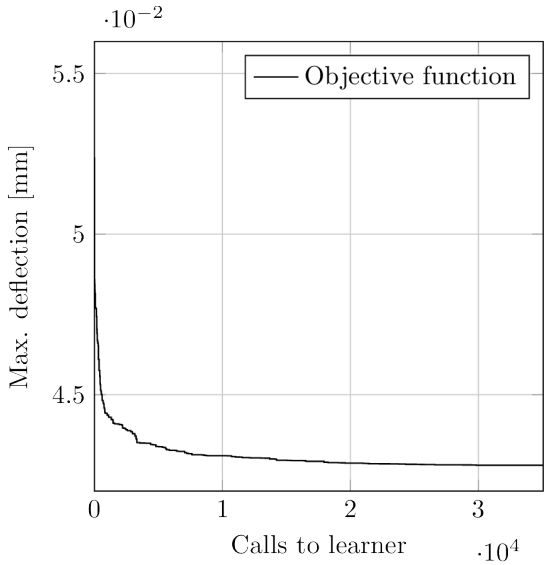
The difficulty involved in the implemented non population based algorithms is that at a given moment within the optimisation procedure, only one defined solution is available. Training on this solution only is not sensible as the learning algorithm would loose its generalization capabilities. Therefore, the implementation of LS, ILS, TS and TS using candidate list solutions use the eager learner without online learner. As the lazy learner learns by adding more samples to its knowledge, it is possible to perform online learning in the same manner as described in Section 6.3.1.2, with the adaption of $N_r = n_b$, i.e. every K steps the optimisation procedure is interrupted, the beams contained in the current solution are calculated using the FE model and the results are added to the knowledge of the lazy learner. The eager learner as well as the lazy learner are pre-trained on the training set described in Section 6.2.1.

6.3.2 Local search

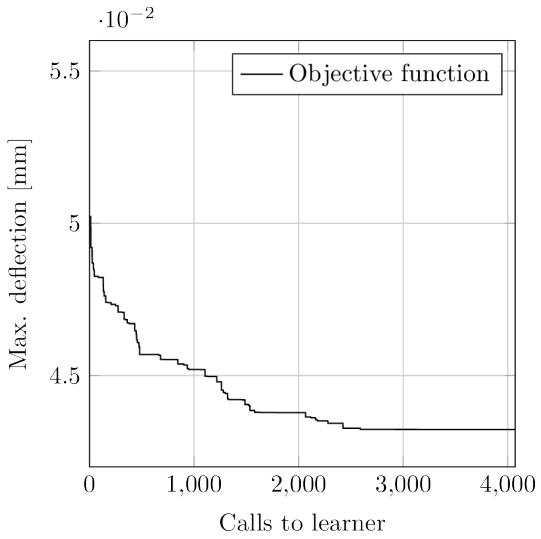
As the purpose of the following sections is the comparison of the optimisation algorithms, online learning is, in favour of faster computation, not considered, since it has, in-between the learning phases, no effect on the algorithm.

The algorithm is at first tested starting from a completely random beam setup, once using the eager learner and once using the lazy learner. The results of those runs are shown in Figure 6.9 where Figure 6.9a shows the result for the eager learner and Figure 6.9b the result for the lazy learner.

The path shown in Figure 6.9b converges, compared to the path shown in Figure 6.9a, at an earlier stage of the optimisation procedure. Furthermore, this is clarified when comparing the deflection of the initial beam setup with the deflection at convergence (both values are obtained using the FE model). The optimiser using the eager learner managed to improve the initial solution by 17 % whereas the one using the lazy leaner improved the solution by 14 %. This suggests that the lazy leaner is not able differentiate between beams at later stages of the algorithm, thus the returned set of nearest neighbours is constant, resulting in equal values for $q_{\max}$. The LS algorithm using the eager learner needed about 50 s to optimise the beam setup, the one using the lazy learner about 31 s. The lazy learner was obviously faster due to the lower



(a) Exemplary optimisation progress resulting from approximating the FE model with the eager learner



(b) Exemplary optimisation progress resulting from approximating the FE model with the lazy learner

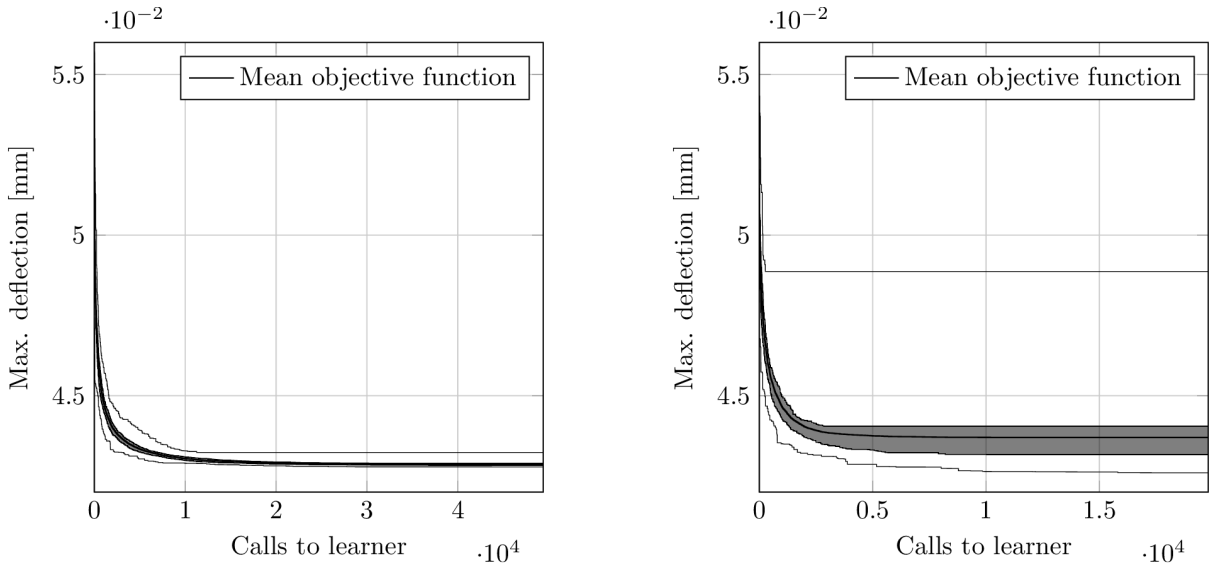
Fig. 6.9: Comparison of two LS runs using different learning algorithms, staring from the same beam setup

number of calls to the objective function. Nevertheless, the ratio between the calls to the eager learner and the lazy leaner is approximately 10, whereas the ratio of the computation times is only 1.6, thus exemplifying the previously discussed disadvantage of lazy learners having to build the metamodel on every request. To verify the observations both algorithms are tested on additional 100 randomly generated beam setups. The results of those runs are shown in Figure 6.10.

Figure 6.10b shows that the previously made assumptions holds for different starting configurations as well. Furthermore, the results amplify the problem of the lazy learner, as the maximum value path is far from the 5 % trimmed mean and even the best obtained results perform poorly compared to the results from the eager learner. This suggests that, in order for the lazy learner to perform as well as the eager learner, periodic updates of the stored samples, with deflections obtained from the FE model, are necessary. Anticipating the discussion in Section 6.4.2 this renders lazy learner for the stated problem useless due to the increased computational effort. Therefore, the lazy learner will not be used in further calculations.

6.3.3 Iterated local search

As well as LS, ILS is tested on a random beam setup. Figure 6.11 shows one exemplary optimisation run performing ILS on the original problem. ILS needed about 16 s to optimise the beams setup, which is, compared to LS, an immense reduction of the computation time. Nevertheless, both algorithms converge against a similar result.



(a) Optimisation progress resulting from approximating the FE model with the eager learner

(b) Optimisation progress resulting from approximating the FE model with the lazy learner

Fig. 6.10: Comparison of two LS runs using different learning algorithms starting from 100 different beam setups

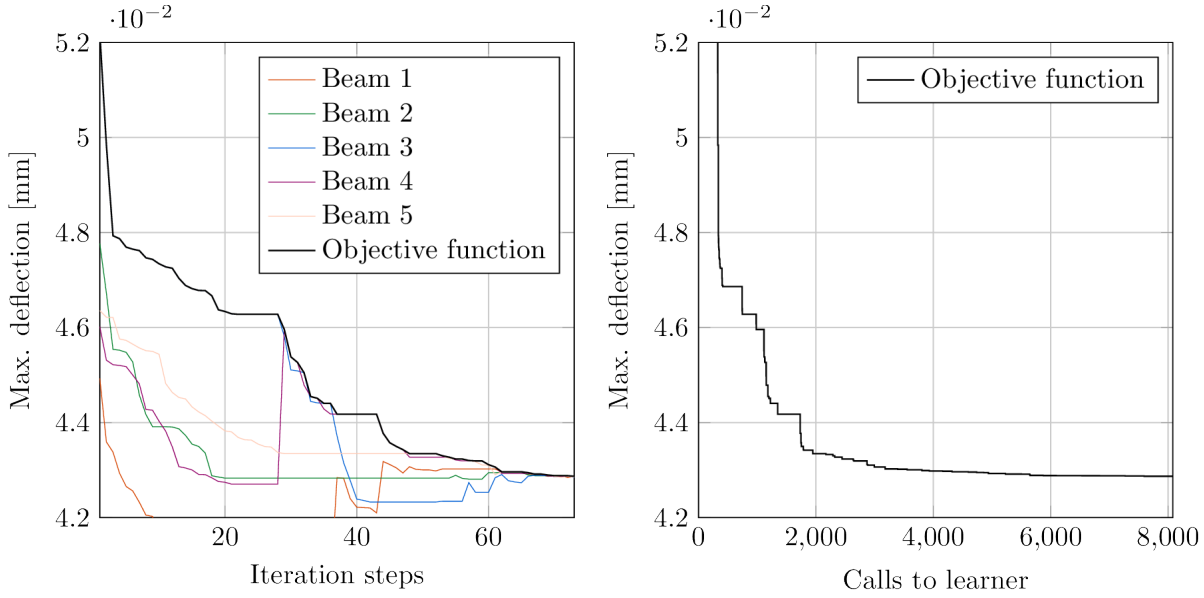


Fig. 6.11: Exemplary optimisation progress of a ILS run for $N_c = 25$ and $\epsilon_c = 0.0001$

6.3.4 Comparison of local search and iterated local search

For comparison LS and ILS are tested on 100 randomly chosen beam setups. The results of those runs are shown in Figure 6.12.

As well as for the simplified problem ILS converges faster than LS and tends to overcome local minima. This assumption is in so far visible as the maximum path of LS in Figure 6.12 shows an early convergence. In Section 5.3 the late increase of the improvement rate of ILS for the simplified problem is discussed. This behaviours is also observable for the optimisation of the original problem.

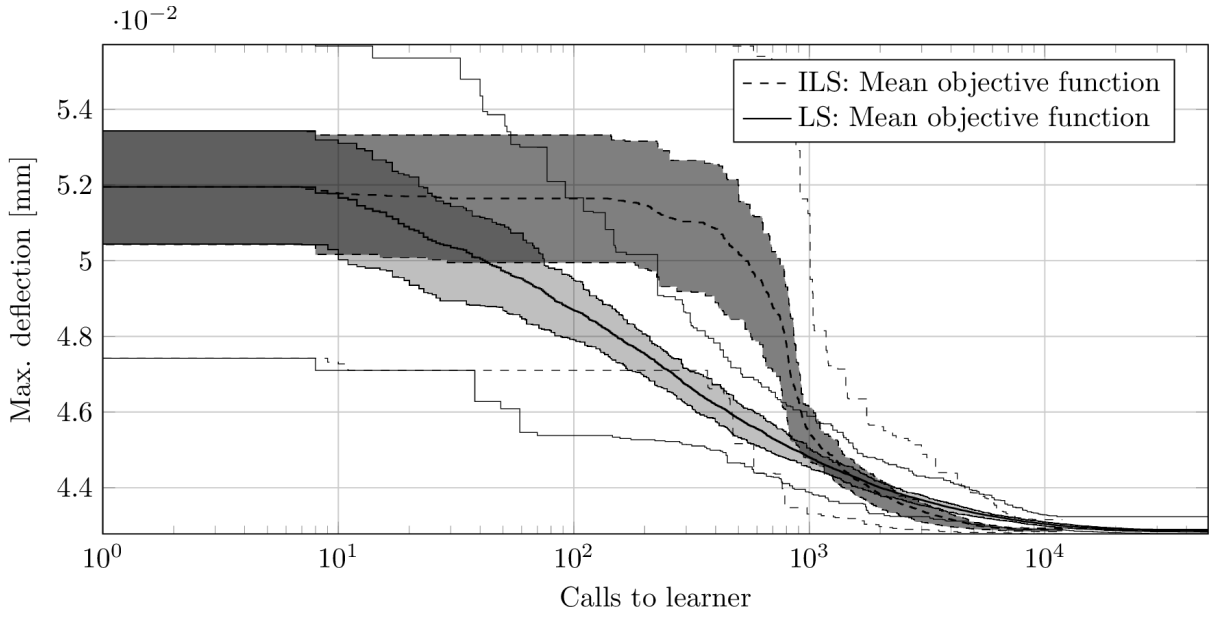


Fig. 6.12: Comparison of 100 LS and ILS runs starting from equal beam setups

6.3.5 Tabu search

Figure 6.13 shows a TS run on the original problem for $N_c = 50$, $N_n = 200$, $N_{T,1} = 50$, $N_{T,2} = 10$ and $N_{T,3} = 50$ using the eager learner. Similar to the simplified problem, in comparison to LS, TS converges slower. The computation time needed for the shown optimisation run was about 140 s.

6.3.6 Tabu search using candidate list strategies

Figure 6.14 shows a TS run using a candidate list solution on the original problem for $N_c = 50$, $N_n = 200$, $N_{T,1} = 50$, $N_{T,2} = 10$, and $N_{T,3} = 50$ using the eager learner. Compared to the original TS algorithm, TS using candidate list strategies converged faster, thus the computation time was reduced to about 67 s. By evaluating $q_{\max}$ for all beams at every stage of the optimisation, the effect of performing meaningful swaps of lamellas between beams is perceptible.

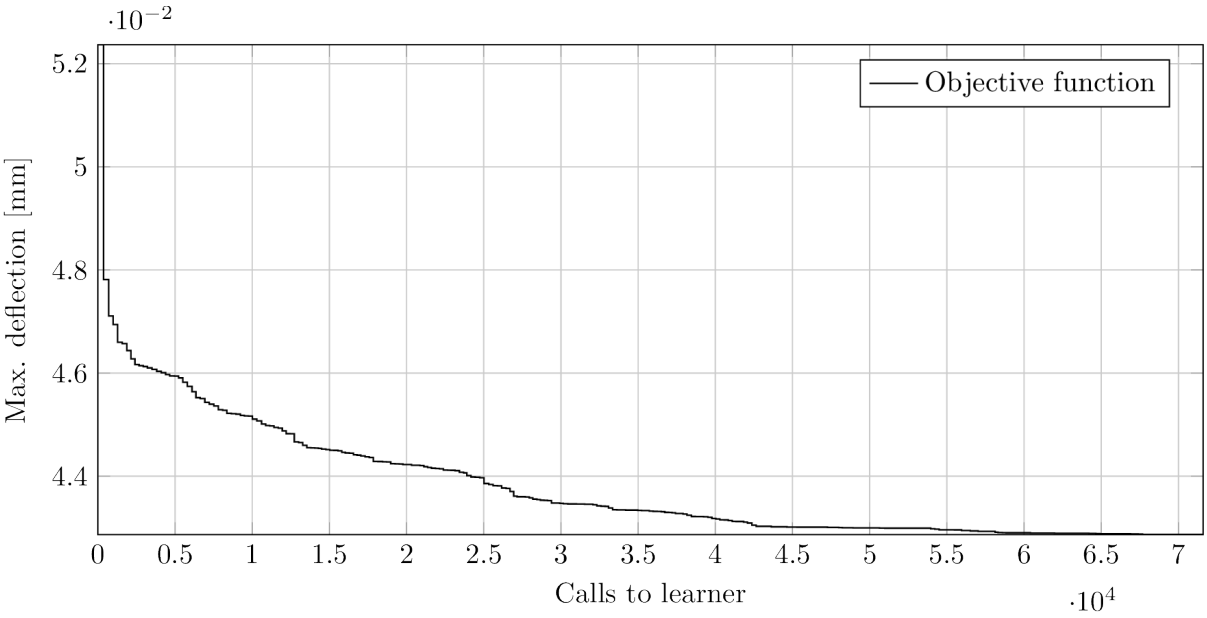


Fig. 6.13: Optimisation progress of a TS algorithm starting with a random beam setup

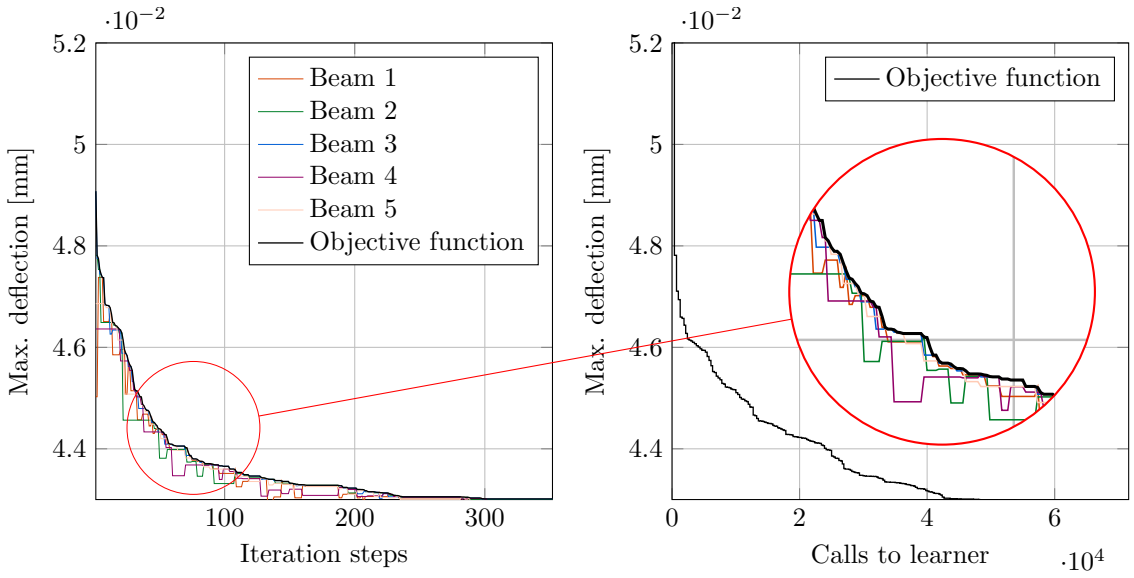


Fig. 6.14: Optimisation progress of a TS algorithm using candidate list solution starting with a random beam setup

6.3.7 Comparison of tabu search with and without using candidate list strategies

For comparison both TS algorithms are tested on 100 randomly chosen beam setups. The results of those runs are shown in Figure 6.15. As well as for ILS, by performing meaningful swaps through candidate list strategies, TS converges faster. Compared to LS and ILS this adaption has a stronger impact and, therefore, the TS algorithm using candidate list strategies prematurely converges. This causes the algorithm to get "trapped" at a solution far from the solution obtained by the original TS algorithm. Furthermore, the effect of premature convergence is noticeable as the maximum value path is far from the 5 % trimmed mean path.

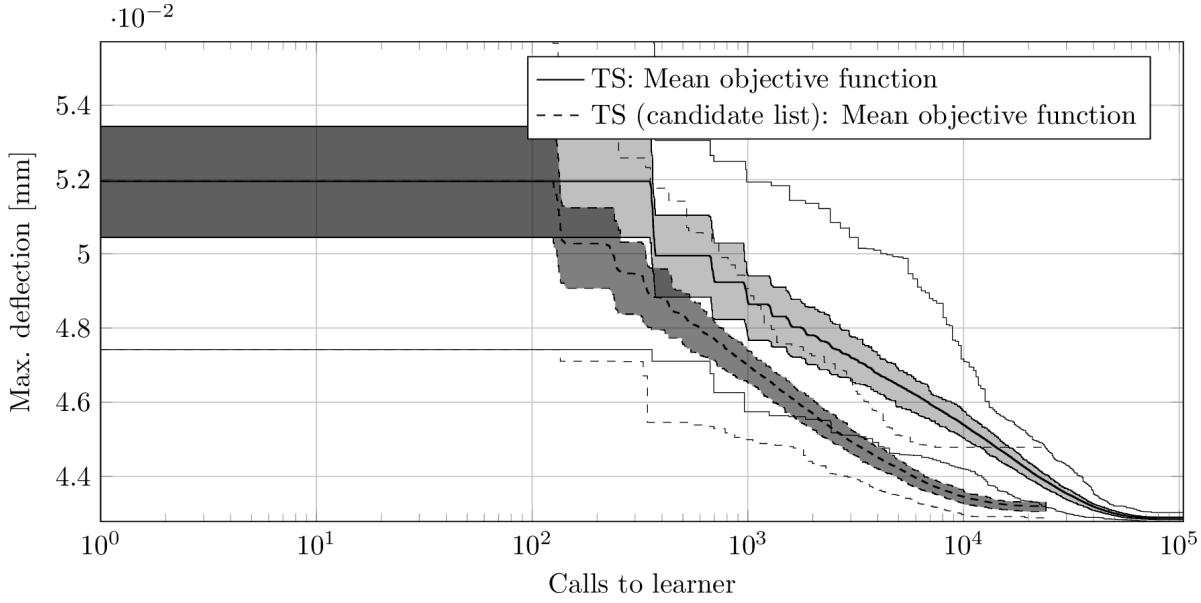


Fig. 6.15: Comparison of 100 TS runs with and without the usage of candidate list solutions

6.3.8 Genetic algorithms

As described in Section 5.8, to overcome the difficulty of finding the optimal parameter combination for the GA, the software package irace [32] was used, to perform automatic parameter tuning. The parameter bounds are, in most parts, derived from Section 5.8. Nevertheless, as stated in Section 4.8.3, the mutation function must be adapted for the original problem. Hence the parameters for irace from Section 5.8 are adapted as follows:

Mutation function : swap two lamellas or flip one, replace with random beam setup

The parameter sweeps are performed on the example from Section 5.1, using no online learning on a number of 250 and 1000 generations. The maximum number of experiments is, in comparison to the simplified problem, reduced to 5000, since optimising the original problem is computationally more expensive.

The best 5 parameter configurations are shown in Table 6.3. Herein the best parameter configurations are those which delivered on average the best results while maintaining a small

inner quartile range, for 250 and 1000 generations. The solution quality for the 5 parameter combinations is shown in Figure 5.8. Similar to the simplified problem the results show a clear preference of the GAs operations. Compared to the parameter combinations for the simplified problem – shown in Table 5.1 – the best results are now achieved by using a completely random initial population, larger mutation rates of up to 100 % and solutions waiving elitism. Interestingly, parameter combination 3 uses a mutation rate of 100 % an no elitism, thus even the best population members are mutated in every generation.

Tab. 6.3: Top 5 parameter combinations for the original problem using the eager learner

Parameter combination:	1	2	3	4	5
Population size N_p	79	89	93	94	100
Mutation rate p_m	0.94	0.94	1.0	0.99	0.99
Crossover rate p_c	0.9	0.55	0.97	0.88	0.86
Elitism ϵ_c	1	1	0	0	3
Crossover function f_c	Partially mapped crossover				
Mutation function f_m	Swap or flip				
Selection function f_s	Tournament selection				
Chromosome type	B				
Initial population	Random initial population				
Tournament size t	10	8	9	9	9

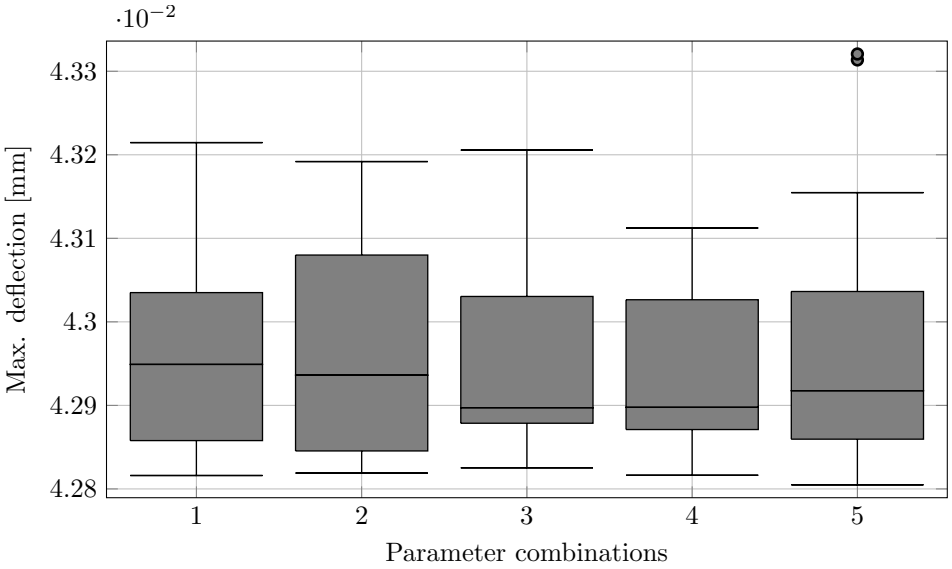


Fig. 6.16: Solution quality for the parameter combinations shown in Table 6.3

Figure 6.17 exemplary shows the optimisation procedure of 5 GA runs using different parameter combinations. The computation time needed for each of the 5 parameter combinations varies between 136 s and 170 s.

Figure 6.18 shows a comparison of the 5 different parameter combinations, starting from 50 different initial populations.

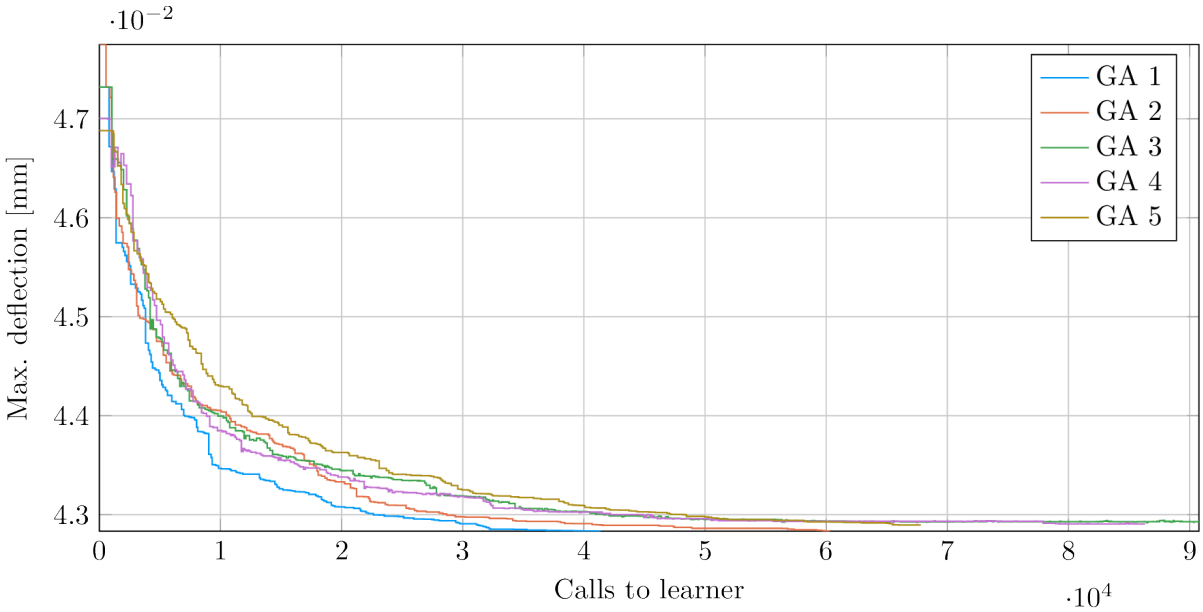
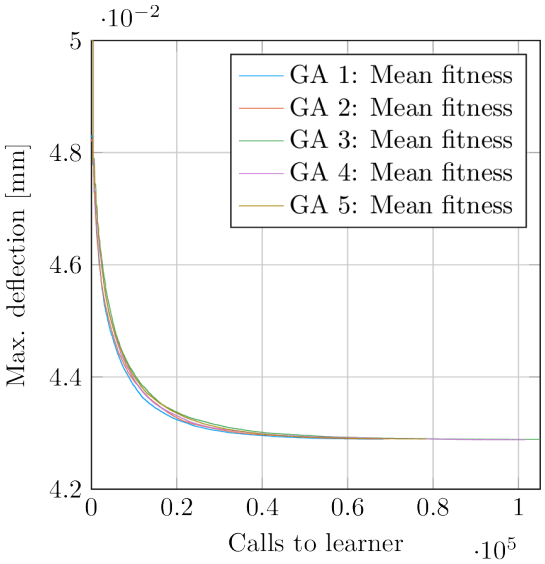
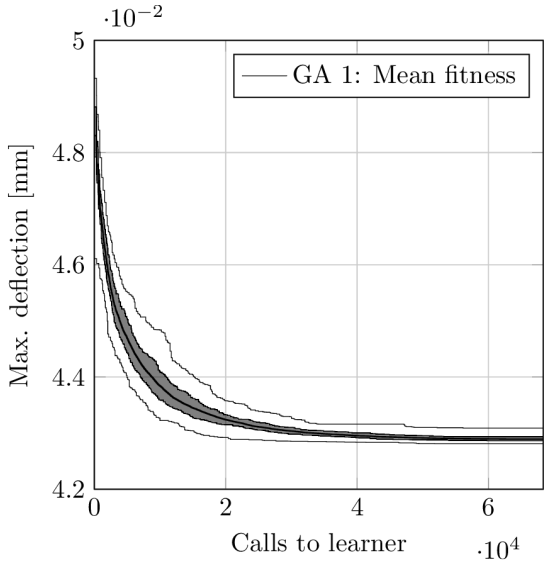


Fig. 6.17: Optimisation progress of the 5 best parameter combinations from Table 5.1

The 5 % trimmed mean value curves of all combinations are shown in Figure 6.18a. As all 5 parameter configurations deviate only marginally from each other, the trend of the optimisation procedure in Figure 6.18b is only shown for parameter configuration 1.



(a) Comparison of the mean value of the fitness function for the top 5 parameter combinations



(b) Optimisation progress of 100 run for parameter combination 1

Fig. 6.18: Comparison of 100 optimisation runs using the top 5 parameter combinations

6.3.9 Comparison of the used metaheuristic algorithms for the non simplified problem

Table 6.4 shows a summary of the results obtained from the MAs tested for the original problem. The 5 % trimmed mean of all discussed algorithms is shown in Figure 6.19. Based on these results, it is possible to draw the following conclusions:

1. TS using candidate list strategies performed – though it converged faster than TS – worst, as it converges against local minima. This is also reflected in a relatively high standard deviation and a large amount of runs which were not able to at least reach the smallest average value of the other algorithms. Therefore, TS using candidate list solution is, in the current implementation, not able to perform as well as the other discussed algorithms. Thus it is not used in further calculations.

2. The best solutions are reached by LS and ILS, albeit the other algorithms performed similar.

3. When comparing the worst solutions of the test runs, the GAs and TS managed to reach the lowest deflection. Similar to the simplified problem, despite LS found the best solution, the runs also contained the worst solutions of all discussed algorithms. This is further reflected in the high standard deviation shown by LS.

4. ILS managed to reach the highest average deflections significantly faster than all other algorithms.

5. The GA using parameter combination 4 outperforms the other GAs, as this combination delivers the lowest standard deviation of all algorithms, the highest best value of all GAs, and performs on average similar to LS. Furthermore, the GA using parameter combination 4 shows the lowest amount of runs which never reach the highest average value. This might root in the slightly higher mutation rates and not using elitism, which provides a broader search field.

Generally, it should be noted that all algorithms, except TS using candidate list solutions, deliver results within a range of 1.0 %, between the best and the generally worst result obtained by LS. This emphasizes the usage of MA for the task of optimising GLT beams.

Based on the above conclusions it is possible to pick two algorithms, applicable to different use cases:

1. ILS should be used when it is important to find good solution in a minimal amount of time. Compared to LS, ILS reaches good solutions faster and is likely to be able to escape local minima. However, compared to GAs and TS the algorithm is less robust.

2. GAs should be used when it is important that the algorithm delivers good solution in every run and when computation time is of secondary importance. Apparently, parameter combination 4 delivers the best results among the other GAs. When using algorithms

Tab. 6.4: Solutions of 100 runs of LS, ILS, TS and TS using candidate list solutions, where $f(\pi^*)$ is used as the objetive function and 50 runs of GAs where $1/f(\pi^*)$ is used as objective function.

	LS	ILS	TS	TS (cand.)	GA 1	GA 2	GA 3	GA 4	GA 5
Best[1]	**4.278**	**4.278**	4.279	4.288	4.281	4.281	4.279	4.281	4.280
Worst[1]	**4.324**	4.316	4.303	**4.478**	4.309	4.304	4.308	4.301	4.307
Avg.[1]	**4.287**	4.289	4.288	4.323	4.290	4.290	4.288	4.288	4.290
Std.[1]	0.008	0.007	0.006	0.029	0.006	0.006	0.006	0.004	0.006
Stepcount for reaching the highest avg. value of 4.29 (Excluding LS using candidate lists)									
Earliest	10001	2824	38123	19217	20698	33076	31593	38136	30735
Avg.	18971	5936	58534	19217	44564	46857	59335	59943	51850
Never[2]	26.0	38.0	33.0	99.0	38.0	42.0	30.0	22.0	38.0

[1] $[10^{-2}\,\text{mm}]$ [2] [%]

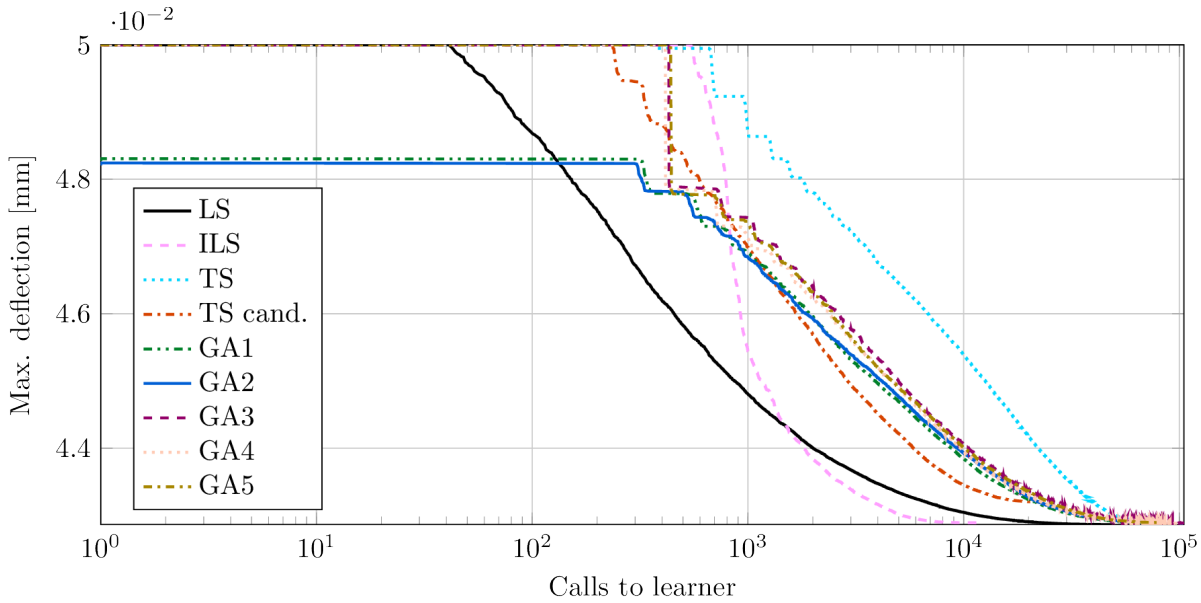


Fig. 6.19: Comparison of the 5 % trimmed mean values of the discussed metaheuristic algorithms

without elitism, it is important to notice that the best solution is not necessarily contained in the last generation.

6.4 Verification of the applicability of online learning

To verify the usefulness of the online learning procedure described in Section 6.3.1, the example from Section 5.1 is optimised using learners, trained on the training set described in Section 6.2.1.

6.4.1 Online learning using the eager learner

As described in Section 6.3.1 the retraining phase for the learner starts after convergence of the optimisation procedure. To be able to measure the success of an optimisation the initial deflection is calculated. Table 6.5 shows the best individual contained in the random initial population. The evaluation of the objective function results in 4.8146×10^{-5} mm.

Tab. 6.5: Comparison of the deflections obtained by the eager learner and the FE model, for the best individual contained in the initial population

Beam	Eager Learner [mm]	FE model [mm]	Loss [mm][1]
1	4.6083×10^{-5}	4.5906×10^{-5}	3.1189×10^{-14}
2	4.5692×10^{-5}	4.6021×10^{-5}	1.0838×10^{-13}
3	4.7319×10^{-5}	4.6070×10^{-5}	1.5587×10^{-12}
4	4.7310×10^{-5}	4.4703×10^{-5}	6.7936×10^{-12}
5	4.7891×10^{-5}	$\mathbf{4.8146 \times 10^{-5}}$	6.5375×10^{-14}

[1] The values are obtained by using the loss function from Section 6.2.4.

The optimisation is performed using parameter set 4 from Table 6.3, resulting in the deflections shown in Table 6.6. Evaluating the objective function for the deflections obtained by the FE model, results in 4.3897×10^{-5} mm. Compared to the initial function values from Table 6.5, the deflection is reduced by 8.83 %.

Tab. 6.6: Comparison of the resulting deflections – after an optimisation run on the practical example (Section 5.1) using the eager learner – with deflections obtained by the FE model

Beam	Eager Learner [mm]	FE model [mm]	Loss [mm][1]
1	4.2849×10^{-5}	4.3099×10^{-5}	6.2141×10^{-14}
2	4.2906×10^{-5}	4.2956×10^{-5}	2.4705×10^{-15}
3	4.2888×10^{-5}	4.2843×10^{-5}	1.9722×10^{-15}
4	4.2898×10^{-5}	4.3246×10^{-5}	1.2087×10^{-13}
5	4.2877×10^{-5}	$\mathbf{4.3897 \times 10^{-5}}$	1.0390×10^{-12}

[1] The values are obtained by using the loss function from Section 6.2.4.

The training set to perform further training is generated by picking the unique beams from the last population of the GA. For the given optimisation run this results in a number of 208 different beams, hence the computationally costly FE model needs to be evaluated for 208 times.

Table 6.7 shows the resulting deflections after the optimisation is performed, using the retrained eager learner. The objective function evaluates to 4.3409×10^{-5} mm. This matches a reduction of 9.84 % compared to the initial value from Table 6.5, hence compared to the result obtained without using online learning, the deflection is further reduced by 1.01 %. However, the computation for reaching this reduction is a lot more costly as the retraining of the learning algorithm involves actually calculating the FE model and, furthermore, optimising the metamodel based on the newly obtained deflections. Hence, online learning helps finding better solution, it is computationally costly.

Tab. 6.7: Comparison of the resulting deflections – after an optimisation run on the practical example (Section 5.1) using the retrained eager learner – with deflections obtained by the FE model

Beam	Eager Learner [mm]	FE model [mm]	Loss [mm][1]
1	4.3194×10^{-5}	$\mathbf{4.3409 \times 10^{-5}}$	4.6065×10^{-14}
2	4.2836×10^{-5}	4.3252×10^{-5}	1.7319×10^{-13}
3	4.2855×10^{-5}	4.2786×10^{-5}	4.6562×10^{-15}
4	4.2944×10^{-5}	4.3040×10^{-5}	9.2099×10^{-15}
5	4.2902×10^{-5}	4.3386×10^{-5}	2.3383×10^{-13}

[1] The values are obtained by using the loss function from Section 6.2.4.

6.4.2 Online learning using the lazy learner

As already mentioned in Section 6.3.2, the problem involved when using the lazy learner is that convergence occurs at an early stage of the optimisation, as the algorithm runs out of differing neighbours. Hence the solution returned consists of the same set of neighbours, though different beams are evaluated. The online learning implementation from Section 6.3.1.2 suggested that the learning phases are performed at every K steps during the optimisation run, on a portion of the current population of size N_r. Initially the lazy learner is trained on the starting population. Figure 6.20 shows the optimisation process using the lazy learner for $K = 10$ and $N_r = 50$.

The requirement of online learning for kNN is apparent in Figure 6.20 as the optimisation progress converges at an early stage. Furthermore, it is visible that by providing the kNN algorithm with new samples, the fitness function changes and the algorithm is able to leave the previous minima. The resulting optimised beam setup, in comparison to the values obtained by the FE model, is shown in Table 6.8. The objective function evaluates to 4.4565×10^{-5} mm, which is a reduction of 7.44 % to the best solution contained in the initial population. Compared to the results obtained through the eager learner, the lazy learner is not even performing as well as the eager learner without online learning. Besides the recurring evaluations of the FE model

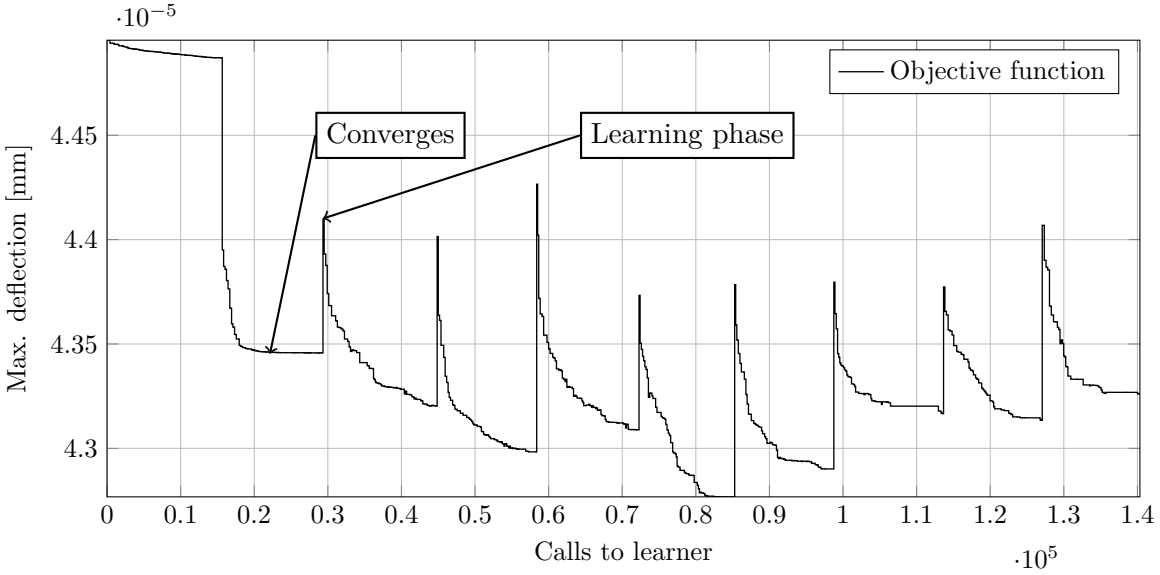


Fig. 6.20: Optimisation procedure for a GA using the lazy learner and online learning.

are computationally intense. Therefore, the lazy learner is, in its present implementation, not applicable to the stated problem and will not be used in further calculations.

Tab. 6.8: Comparison of the resulting deflections – after an optimisation run on the practical example (Section 5.1) using online learning and a lazy learner – with deflections obtained by the FE model

Beam	Eager Learner [mm]	FE model [mm]	Loss [mm][1]
1	4.3935×10^{-5}	4.3865×10^{-5}	5.0298×10^{-15}
2	4.3590×10^{-5}	4.3618×10^{-5}	7.7284×10^{-16}
3	4.4429×10^{-5}	4.3792×10^{-5}	4.0567×10^{-13}
4	4.4943×10^{-5}	$\mathbf{4.4565 \times 10^{-5}}$	1.4285×10^{-13}
5	4.4162×10^{-5}	4.4119×10^{-5}	1.8624×10^{-15}

[1] The values are obtained by using the loss function from Section 6.2.4.

6.5 Effects of variations in the optimisation problem

This section examines the effects of variations in the definition of the optimisation problem and gives an insight on how much improvement of the load bearing behaviour is possible. Furthermore, the effects of using lamellas from different grading classes to build combined GLT beams are investigated.

In the following sections the term GA refer to a GA using parameter combination 4 from Table 6.3 and calculated deflections always refer to the results obtained from the FE model. This applies unless otherwise specified.

6.5.1 Quantification of a range of possible improvement for the practical example

Up until now the performance of an optimisation was always measured against the initial solution. As the initial solution is almost entirely random[1], the performance depends on how "good" the initial guess of the solution was.

In Section 6.3.2 and Section 6.4.1 a performance measure of this kind was applicable as only performances based on equal initial values were compared. To be able to determine a possible range of improvement, the objective function from Equation (4.1) shall be – instead of being minimised – maximised, hence resulting in a π^* containing the beam with the maximum possible deflection. The optimisation is performed by using the GA on the example from Section 5.1.

Instead of optimising the inverse of Equation (4.1), Equation (4.1) can directly be used as fitness function.

The optimisation process was stopped after 2000 generations. The progress shown in Figure 6.21 clearly illustrates that the algorithm converged.

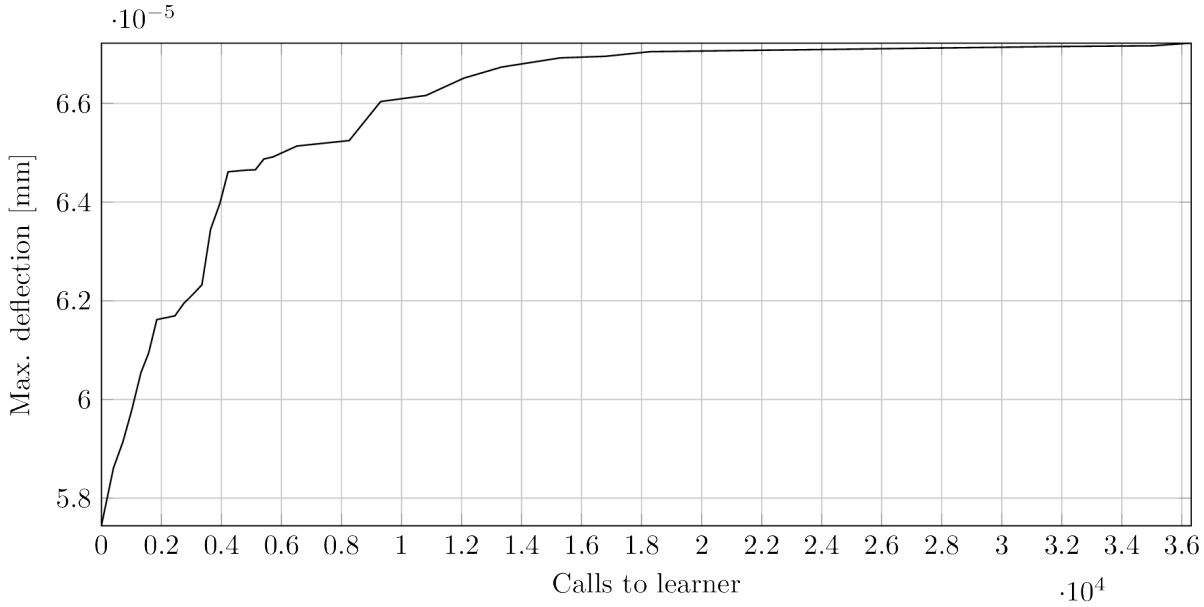


Fig. 6.21: Optimisation progress for finding the worst possible solution for Equation (4.1) for the practical example

The "best" solution results in a deflection of 6.1658×10^{-5} mm. Hence compared to the results from Section 6.4.1 of

- 4.3897×10^{-5} mm without online learning and
- 4.3409×10^{-5} mm with online learning,

the actual possible improvements are

- 28.81 % and

[1] See Section 4.8.6 for a discussion on pure random and pseudo random initial populations.

- 29.60 %.

Thus, the improvement by using online learning is reduced to 0.79 %.

Commonly, GLT beams are constructed entirely random – besides the construction of combined GLT beams, as discussed in Section 1.1. Therefore, to assess the possible improvement compared to the common practice, 1000 random beams, constructed from the 50 lamellas, are calculated using the FE model. On average the worst resulting beam has a deflection of 5.176×10^{-5} mm. The coefficient of variation for the tested sample is 0.035 Thus, the actual possible improvements, compared to the randomly generated beams are

- 15.19 % and
- 16.13 %.

6.5.2 Determining the effect of combined sets of LS15 and LS22

As describe in Chapter 1 it is common practice to optimise the material usage for GLT beams by combining lamellas of different grading classes in one beam.

To determine how "infecting" a set of lamellas of a higher grading class (LS22), with lamellas from a lower grading classes (LS15), changes the outcome of the optimisation, a new set of lamellas for the practical example is generated.

Table 6.9 shows the lamellas from [47] used for this test set.

The column *ID* contains the frequently mentioned lamella ID. The column *Nr.*, combined with the grading class from column *GC*, allow the identification of a lamella in the experiments from Serrano et al. [47].

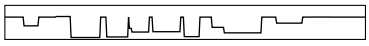
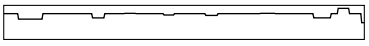
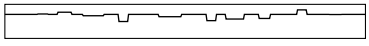
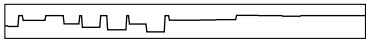
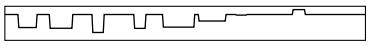
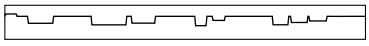
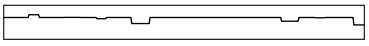
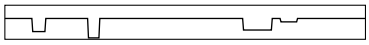
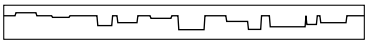
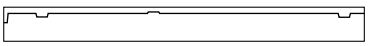
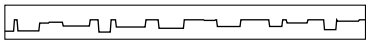
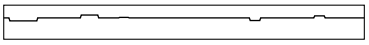
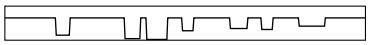
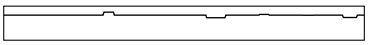
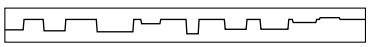
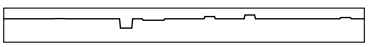
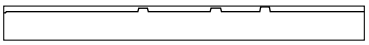
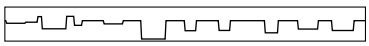
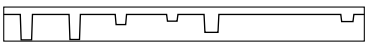
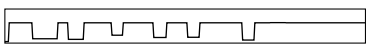
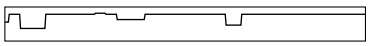
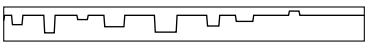
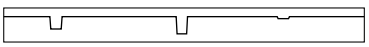
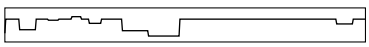
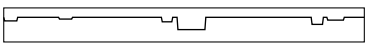
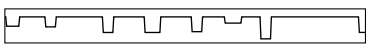
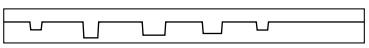
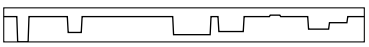
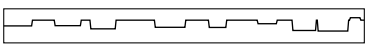
Column $E(x)$ contains the longitudinal stiffness profile of each lamella. All profiles within Table 6.9 are equally scaled to be comparable.

Lamellas 1–20 are picked from grading class LS15, lamellas 21–50 from grading class LS22. When comparing the longitudinal stiffness profiles of both grading classes, it is obvious that the lamellas from LS15 have overall lower values for $E(x)$ and a larger number of weaknesses i.e. knots.

Based on the influence of weaknesses at certain locations within a beam, derived in Section 6.2.3, the expected outcome when optimising the given lamellas is, that lamellas of grading class LS15 end up in areas of less influence. This assumption matches the common practice when building combined GLT beams, as usually the outer layers of the beam are assembled with lamellas of higher grading class than the inner layers.

As described in Section 6.5.2, to be able to quantify the performance of the optimisation, at first the worst solution is considered. By using a GA the largest possible deflection is 7.7578×10^{-5} mm. Furthermore, to obtain the improvement compared to a beam constructed according to common practice 1000 random beams are generated. Herein, in order to consider that combined GLT beams are build with stronger lamellas in the outer layers, the beams are constructed such that the outer two layer on each side of the beam only consist of lamellas of grading class LS22. The resulting beams have a average deflection of 5.4958×10^{-5} mm with a coefficient of variation of

Tab. 6.9: Lamellas used in the optimisation problem

ID	GC	Nr.	$E(x)$	ID	GC	Nr.	$E(x)$
01	LS15	43		26	LS22	85	
02	LS15	44		27	LS22	86	
03	LS15	45		28	LS22	87	
04	LS15	46		29	LS22	88	
05	LS15	47		30	LS22	89	
06	LS15	48		31	LS22	90	
07	LS15	49		32	LS22	91	
08	LS15	50		33	LS22	92	
09	LS15	51		34	LS22	93	
10	LS15	52		35	LS22	94	
11	LS15	53		36	LS22	95	
12	LS15	54		37	LS22	96	
13	LS15	55		38	LS22	97	
14	LS15	56		39	LS22	98	
15	LS15	57		40	LS22	99	
16	LS15	58		41	LS22	100	
17	LS15	59		42	LS22	101	
18	LS15	60		43	LS22	102	
19	LS15	61		44	LS22	103	
20	LS15	62		45	LS22	104	
21	LS22	80		46	LS22	105	
22	LS22	81		47	LS22	106	
23	LS22	82		48	LS22	107	
24	LS22	83		49	LS22	108	
25	LS22	84		50	LS22	109	

$E(x)$ is equally scaled for every lamella

0.04. The actual optimisation task is performed using ILS – as described in Section 6.3.3 – and a GA.

Figure 6.22 shows a comparison of both used algorithms. ILS returns a solution which gives a deflection of 4.5798×10^{-5} mm, the GA gives a deflection of 4.5076×10^{-5} mm.

Compared to the worst value, it is possible to improve the solution by

- 40.92 % using ILS and
- 41.85 % using the GA,

whereas compared to the average random solution by

- 16.67 % using ILS and
- 17.98 % using the GA.

This result substantiates that, obviously by using lamella sets with diverse lamella qualities, the possible range of improvement is larger compared to homogeneous sets like the one used for the practical example. Furthermore, it is remarkable that ILS only differs by 0.93 % from the result obtained by the GA, though ILS only needs 4312 calls to the objective function, where as the GA needs 137168.

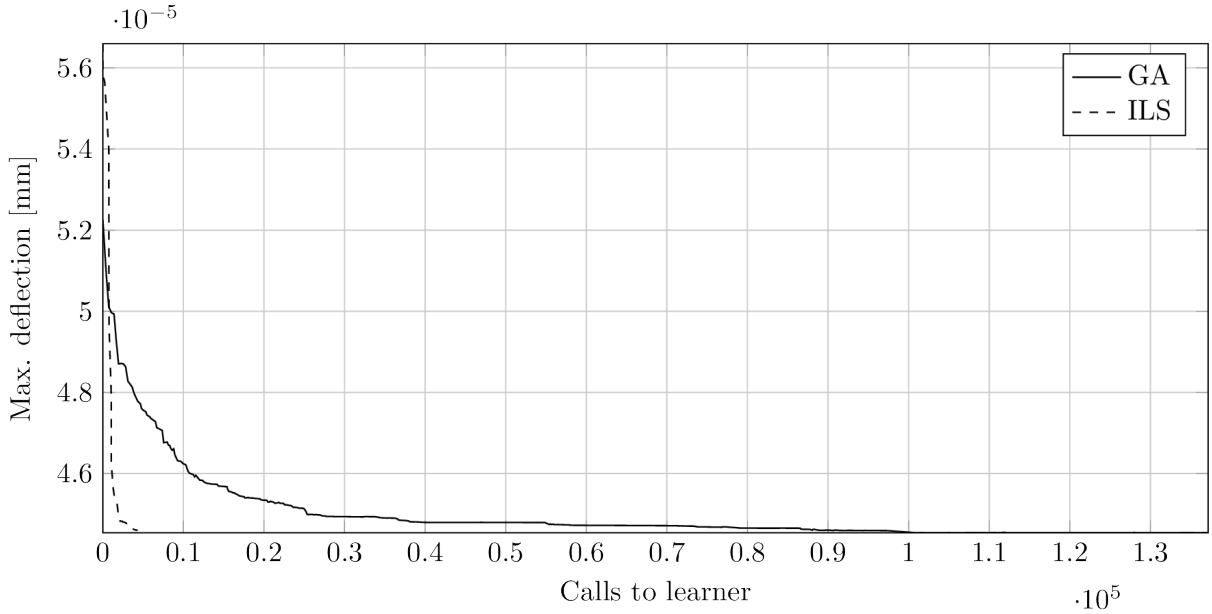


Fig. 6.22: Comparison of ILS and GA for the optimisation problem using different grading classes

The final solution obtained from the GA is shown in Table 6.10.

A visualization of the longitudinal stiffness profile $E(x)$ for every beam contained in this solution is shown in Figure 6.23a.

When looking at the marked area **A** in beam 1 it is apparent how the optimisation algorithm manages to gather "weak" areas close to the beams centre of mass. Furthermore, when examining

Tab. 6.10: Near optimal beam setup for the optimisation problem with combined lamellas from grading classes LS15 and LS22

Location[1]	Beam 1	Beam 2	Beam 3	Beam 4	Beam 5
1	16*	21	28	32	36*
2	14*	33	47	37*	35*
3	03	46	05*	18	43
4	17	25*	31	23*	13
5	20	11*	22*	44*	49*
6	01	12*	38*	09*	07
7	10	27	08	48	45*
8	04	50	06	39	19
9	24*	15	02*	30*	29
10	41	26	40	42*	34*
Deflection [10^{-5}mm]	4.4753	4.4575	4.5067	4.5076	4.4642

Lamella IDs marked with an asterisk are flipped
[1] The location is defined from top to bottom of the beam.

the clearwood areas from beam 3 or beam 5, a colour gradient from light grey to darker grey is noticeable, depicting the decreasing longitudinal stiffness $E(x)$ towards the inner layers.

Figure 6.23b shows the locations of lamellas from grading class LS15 in the final solution. Against the initial assumption, the best solution is not obtained by placing those weaker lamellas on the inner layers. The actual result is quite contrary as beams 1, 2, and 3 have lamellas from grading class LS15 within the outer two layers. Furthermore, beam 1 is almost entirely build from grading class LS15, without being – referring to Table 6.10 – the beam with the highest deflection.

6.5.3 Determining the effect of an increased complexity

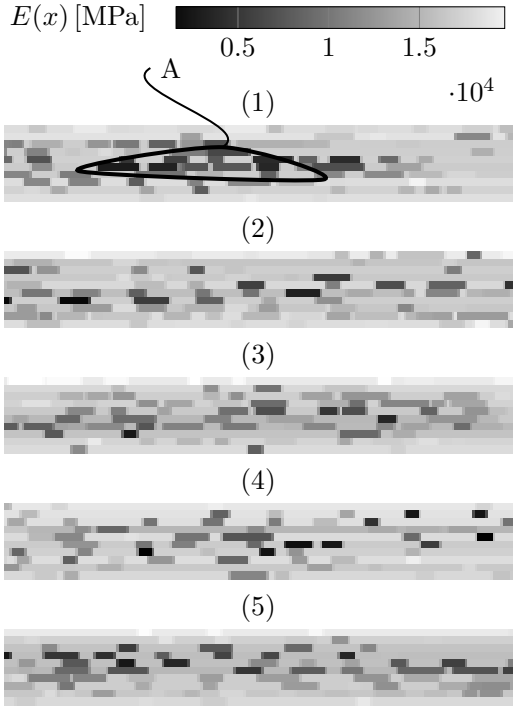
To be able to tell the applicability of the described optimisation algorithms for more complex problems, the number of used lamellas is doubled. The optimisation is performed

1. for $n_l = 100$, $n_{n,l} = 10$, $n_b = 10$ and
2. for $n_l = 100$, $n_{n,l} = 5$, $n_b = 20$,

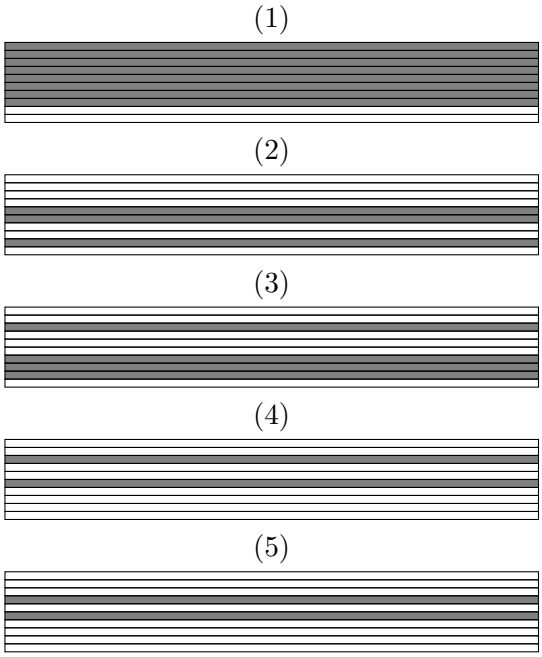
to figure out how the number of lamellas per beam affects the performance of the optimisation.

For both examples the geometry described in Section 5.1 is used, as no requirements for the beam height were made. Of course, for the second example the FE-mesh and the metamodel for the eager learner are updated accordingly.

The lamellas used for the examples are 41–94, 96–135 and 142–147 of grading class LS15. The selection was only refined to exclude lamellas with incomplete stiffness profiles.



(a) Heatmap of $E(x)$ for every beam contained in the best solution. Area **A** clearly shows how "weak" regions are gather close to the centre of mass.



(b) Beams contained in the best solution with highlighted locations of lamellas from grading class LS15

Fig. 6.23: Resulting beam setup from the optimisation using a GA

6.5.3.1 Increased complexity using 10 lamellas per beam

Initially the worst value for the given setup is obtained using a GA. The resulting solution has a deflection of 9.4077×10^{-5}mm. On average, the resulting deflection of 1000 random combinations is 6.9738×10^{-5}mm with a coefficient of variation of 0.047.

The actual optimisation is performed using ILS and the GA. The solutions resulting from the optimisers have a deflection of

- 5.3559×10^{-5} mm for ILS and
- 5.4261×10^{-5} mm for the GA.

The improvement compared to the worst solution is

- 43.07 % for ILS and
- 42.32 % for the GA,

the improvement compared to the average of the random combinations is

- 23.2 % for ILS and
- 22.2 % for the GA.

Seemingly ILS performed better than the GA, though when looking at the deflections obtained by the metamodel of

- 5.1633×10^{-5} for ILS and
- 5.1568×10^{-5} for the GA.

it is apparent that the GA still outperformed ILS, but due to the inexact representation of the metamodel, the deflection obtained from the FE model is coincidentally lower when using the solution obtained from ILS.

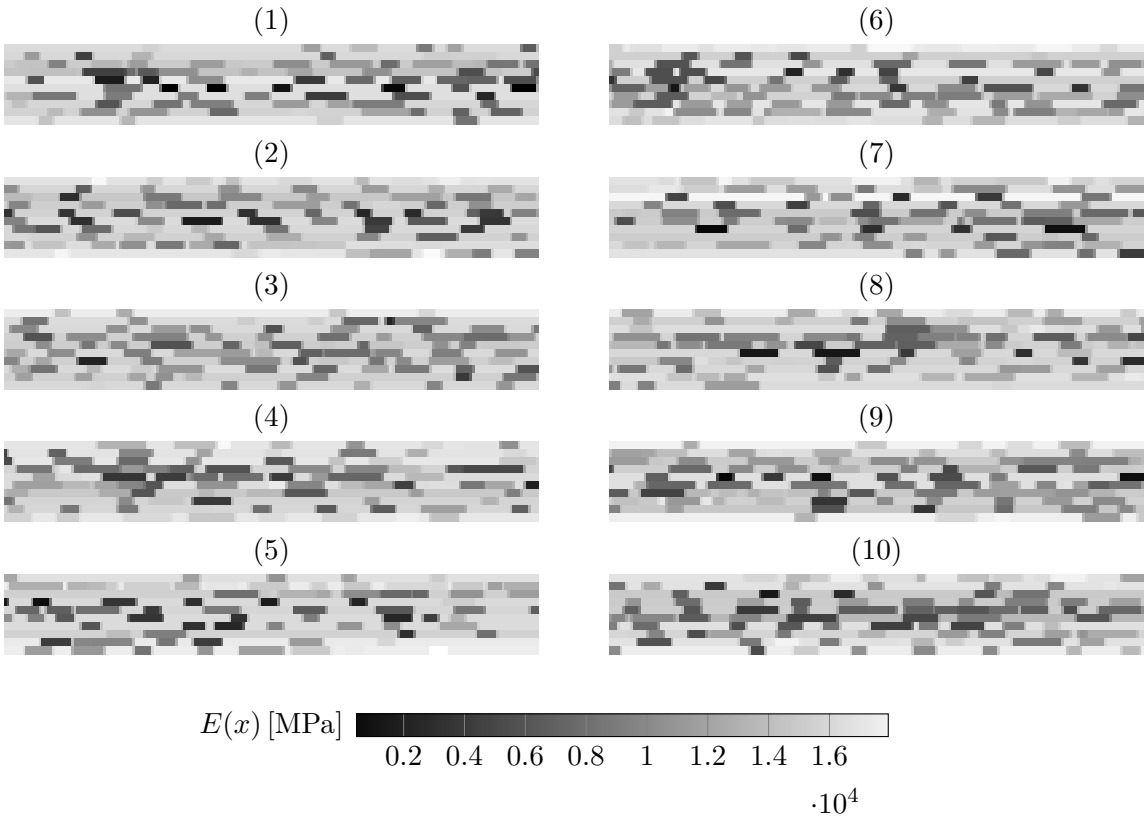


Fig. 6.24: Near optimal solution for the optimisation problem with increased complexity.

Figure 6.24 shows the resulting distribution of $E(x)$ in the beams. Again it is possible to spot areas within the inner layers where weaknesses are gathered.

Figure 6.25 shows the trace of the optimisation procedure for both the GA and LS. It is clearly visible that both algorithms converged. ILS needed 11083 calls to the objective function. Compared to the results from Section 6.5.2, this is an increase of 157.03 %.

As the GA stops after a certain generation, comparing the final number of calls to the objective function is not sensible. More significant is a comparison of the points at which the deflection path converged, shown in Figure 6.22 and Figure 6.25. This comparison suggests that the effort for the GA is about doubled.

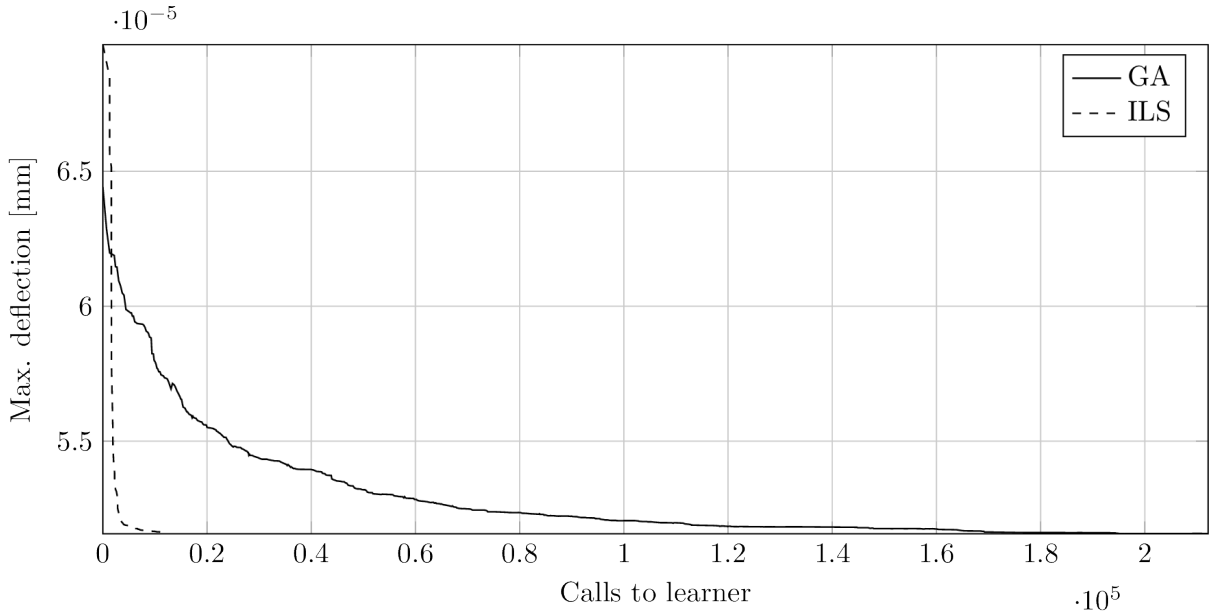


Fig. 6.25: Comparison of ILS and GA for the optimisation problem with increased complexity

6.5.3.2 Increased complexity using 5 lamellas per beam

The optimisation for 5 lamellas per beam is performed in the same manner as described in Section 6.5.3.1.

However, one significant difference occurred. ILS now converges after 3833 steps, which is even faster than for the problem using 50 lamellas from Section 6.5.2. The GA converged after about 1.6×10^5 which is at least better than for the example using 10 lamellas per beam.

The latter is obvious when specializing Equation (1.2) for $n_l = n_b \cdot n_{b,l}$ as

$$\frac{2^{n_b \cdot n_{b,l}} \cdot n_l!}{n_b!}. \tag{6.13}$$

When increasing the number of beams n_b while preserving $n_l = n_b \cdot n_{b,l}$ the numerator is fixed where as the denominator increases factorially, hence reducing the complexity of the problem.

The fast convergence of ILS might be a result of the beam focused optimisation, as the complexity of optimising a single beam was significantly reduced.

6.5.4 Optimisation with removal of bad lamellas

Up until this point the optimisation task was always bound to $n_l = n_b \cdot n_{b,l}$. As stated in Section 1.2 another interesting task is to optimise the arrangement for $n_l > n_b \cdot n_{b,l}$.

This adaption allows the algorithm to sort out “bad” lamellas, which are of non use when building GLT beams.

To be able to cope with this task, the described implementations of the MAs from Section 6.3 need to be adapted. As ILS and GAs prove the perform well for the given task, the changes are only made for those two algorithms.

6.5.4.1 Adaption of the genetic algorithm for sorting out bad lamellas

During crossover, selection, and mutation the GA is unaware of the existence of beams since the chromosome equals π^*. To allow the algorithm to utilize lamellas that will not be build into a beam, π^* must contain all lamellas. Hence the initial definition of π^*, given in Section 4.1 remains intact for $n_l > n_b \cdot n_{b,l}$.

In Section 4.8 two additional states of the algorithm are mentioned, namely the generation of the initial population and the chromosomes repairing phase.

The initial population is dependent on the generated beams in so far as when generating a pseudo random initial population the locations within the beams are recorded. However, as the parameter sweep from Section 6.3.8 clearly favours a pure random initial population, this phase needs no adaption as well.

The chromosome repair on the other hand entirely depends on the resulting beams and should be able to neglect the order of unused lamellas. Thus, the used implementation is not applicable.

As shown π^* remains as defined, hence the remaining unverified part of Equation (4.1) is Φ. The parameter sweep from Section 6.3.8 favoured a chromosome representation without considering building block, hence further only Equation (4.3) will be used.

As Equation (4.3) is defined in terms of $n_{b,l}$ and under the assumption that every beam contains an equal number of lamellas, it is possible to generate L_i^* for all b_i independent of n_l. Furthermore, the definition of Φ in Equation (4.4) is made in terms of n_b, hence it also holds for $n_l > n_b \cdot n_{b,l}$.

Additionally, this defines the arrangement of the chromosome, as all lamellas at positions larger than $n_b \cdot n_{b,l}$ are not assigned to a beam. By implication this means the first $n_b \cdot n_{b,l}$ places in the chromosome define the resulting beams.

These findings leave the need to solely adapt the chromosome repair function.

6.5.4.2 Chromosome repair for sorting out bad lamellas

The implementation described in Section 4.8.5 performs the following steps to repair a chromosome:

1. Use Φ to generate a vector of beams from the chromosome.
2. Sort the vector of beams by the id of the top most lamella in every beam.
3. Reassemble the chromosome based on the sorted vector of beams.

The flaw in context of this work flow is, that during step 1 the unused lamellas are discarded.

Therefore, the unused lamellas need to be considered during the reassembly and furthermore, as stated in Section 6.5.4.1, as well repaired in form of sorting them based on the lamella ID. This ensures that during crossover equal parent chromosomes result in equal child chromosomes.

Hence the previous work flow is extended as follows.

4. Sort the unused lamellas based on the lamella ID.

5. Attach the sorted, unused lamellas to the reassembled, incomplete chromosome from step 3.

6.5.4.3 Adaption of the iterated local search algorithm for sorting out bad lamellas

As described in Section 5.3, ILS optimises beams on different scopes, namely local to the beam and between two defined beams.

As the introduced change of $n_l > n_b \cdot n_{b,l}$ does not affect $n_{b,l}$, the local optimisation procedure is inherited as defined.

The extension takes place on the level of the interchange of lamellas between beams, as not only swaps between two beams, but also swaps between one beam and the set of unused lamellas shall be performed.

In accordance to the suggested implementation it is viable to assume that the unused lamellas build a beam as well. Hence it is possible to apply the neighbourhood definition Γ' from Section 5.3.

However, a swap between a beam and the set of unused lamellas is not made under the same restriction. Therefore, the objective function f' from Equation (4.17) must be adapted as follows.

$$f' : \begin{cases} 0, & \text{if } f(s'_{k,i}) < f(s_k) \\ 1, & \text{if } s'_{k,i} = s_k \text{ and } s'_{m,i} = s_m \\ 2, & \text{otherwise} \end{cases} \tag{6.14}$$

This leads to the effect that a perturbation is only accepted if the actual beam, involved in the swap, is better than the previously worst beam. In comparison the restriction of not worsening beam b_m or solution s_m is omitted as in this case b_m is not part of the actual solution. For interchange of lamellas between actual beams, Equation (4.17) can still be used.

The algorithm is endowed with a second convergence criterion, which ensures that after a number of unsuccessful tries of swapping lamellas with the set of currently not used ones, the focus is set on optimising the actual beams.

6.5.4.4 Results of the optimisation for sorting out bad lamellas

The optimisation is performed using ILS and the GA. Both algorithms are tested on the example described in Section 6.5.2 for $n_b = 4$ and $n_{b,l} = 10$. Therefore, a number of 10 lamellas can be removed from the final solution. Figure 6.26 shows the trace of both optimisation processes.

For the reformulated problem the GA converged a lot faster, whereas ILS needed longer. The performance reduction of ILS might root in the fact that swaps with the set of unused lamellas become less improving over time as the number of "bad" lamellas increases. Furthermore, the number of times performing swaps with better beams is reduced as the number of possible swap locations is increased.

Nevertheless, both algorithms return the following similar deflections of

- 4.3806×10^{-5} mm for ILS and

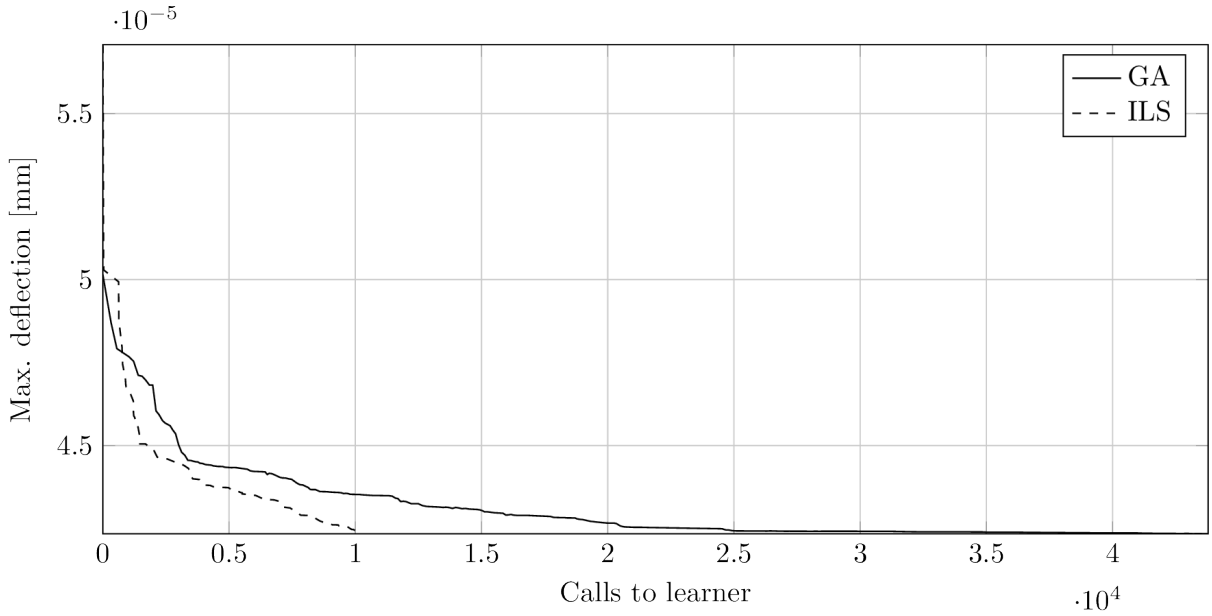


Fig. 6.26: Comparison of ILS and GA using parameter set 4 for the optimisation problem of sorting out bad lamellas

- 4.3514×10^{-5} mm for the GA.

During the optimisation procedure the following lamellas are removed from the final solution.

ILS: 1,7,8,12,20,23,38,44,45,49

GA: 1,7,8,9,13,20,22,38,44,49

Both list obviously correspond very well. Compared to the solution for $n_b = 5$ in Section 6.5.2, mainly lamellas from the inner layers are remove. Figure 6.27 shows the locations where the removed lamellas were build into the beam in the final solution from Section 6.5.2. Lamellas highlighted in light grey are removed by ILS, lamellas highlighted in dark grey by the GA. The remainder is removed by both algorithms.

Figure 6.27 clearly shows the correlation between putting weak lamellas into the inner layers and the adapted algorithms being able to spot those weak lamellas.

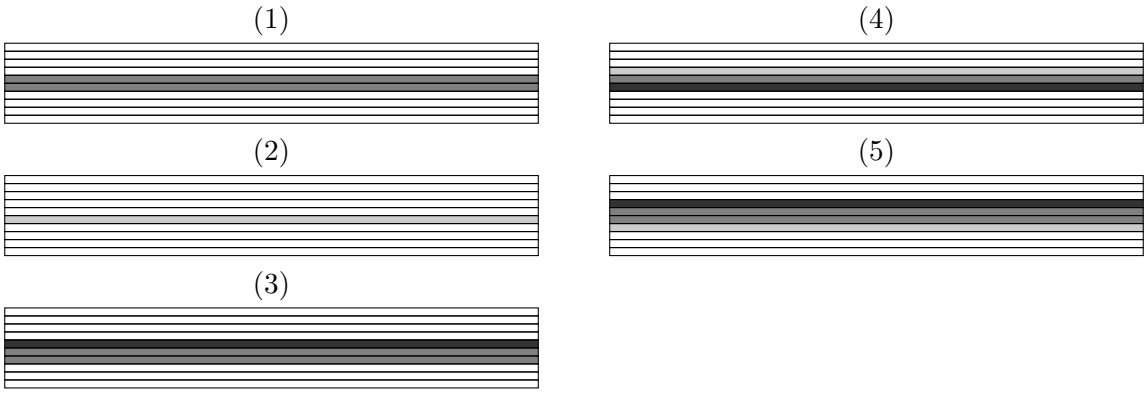


Fig. 6.27: Locations of lamellas removed during the optimisation procedure in the near optimal solution of Section 6.5.2. Light grey lamellas are removed by ILS, dark grey ones by the GA. The remaining ones are removed by both algorithms.

Chapter 7

Summary & conclusion

Chapter 1 gives an introduction to the problem and discusses the complexity of the optimisation task. In Chapter 2, a general overview on the current state of the art for timber board material models was given. Chapter 3 introduces the concept of metaheuristic algorithms, as a possible tool for solving hard optimisation tasks. In Chapter 4 a general, abstract definition of the problem and its components is derived, to allow applicability almost independently of the optimisation or search scheme. Furthermore, the implementation of LS, ILS, TS, and GAs is described for a simple model as well as for a more complex problem (the original problem proposed in Chapter 1).

In Chapter 5 the discussesed MAs are applied to the simplified problem. At the beginning of Chapter 6 the implementation of a FE model for performing the optimisation task is discussed, raising the issue of having a computationally costly calculation model. Thus, two learning algorithms are introduced, that are capable of estimating the results of the FE model. Based on these learning algorithms the original problem is solved and validated. The chapter closes with various adaptations of optimisation tasks for the described problem.

Based on the obtained results the following conclusions can be drawn:

- The task of reducing the deflection of multiple beams through rearranging the lamellas is not, even for the simplified problem addressed in Section 4.2, easily solvable, since the lamellas used to form a beam cannot be seen as individuals. Furthermore, in order to achieve the objective to minimise the maximum deflection of all beams, lamellas might be placed at unobvious positions, where their potential is not fully exploited. Therefore, to obtain good solutions for the optimisation problem, in a reasonable time frame, metaheuristic algorithms are needed.

- The introduced metaheuristic algorithms prove to be applicable for performing combinatorial optimisation tasks for GLT beams and are capable to find near optimal solutions to the stated problem. Nevertheless, ILS and GA managed to outperformed the others as specialist in their domain:
  - ILS is able to find solutions to the stated problem with a minimum amount of objective function evaluations. However, the solution quality strongly depends on the initial solution.
  - GAs found some of the best solutions for the problem, yet needing almost 10 times more evaluations of the objective function than ILS. Nevertheless, they also prove

to be very robust, thus delivering very good solutions, independently of the initial population.

- Both used learning algorithms are able to estimate the maximum deflection of a beam, after training on results from the FE model, very well. Nevertheless, the lazy learner using kNN methods strongly depends on periodic updates of its stored samples – with newly generated results from the FE model – thus, requiring costly evaluations during the optimisation procedure.

  The discussed eager learner, utilising a metamodel in form of linear polynomial regression, performs well, even without updating the model during runtime, therefore being favoured in place of the lazy learner.

- The range of possible improvement depends on the quality of the lamellas i.e. the variability and value of $E(x)$ of the lamellas. Compared to the worst possible beam, built from lamellas subject to the optimisation, the possible improvement is larger when dealing with lamellas of grading class LS15 than when dealing with lamellas of grading class LS22. This is in so far reasonable, as lamellas from grading class LS15 have a larger variability in $E(x)$, thus the algorithm can construct more diverging solutions through precisely orienting lamellas in a way that weaknesses are concentrated in specific areas of the beam (with high or low impact depending on the desired outcome, see Section 6.2.3).

  For the discussed examples in Chapter 6 the possible improvement, compared to how GLT beams and combined GLT beams are commonly produced, ranges from about 15 % – 20 %.

- It is even possible to improve GLT beams assembled under consideration of placing lamellas of higher grading classes in the outer layers of the beam, as during grading the lamella is not classified with respect to its actual usage. This is shown in Section 6.5.2 as against the initial assumption of placing lamellas of a lower grading class within the inner layers of the beam, they are also used in the beam's outer layers.

The scope of this thesis was to find and validate optimisation schemes for the task of optimising GLT beams. Bases on this work, the following extensions and variations regarding the optimisation task and the algorithms are conceivable:

- Throughout this work the structural system of a two point bending test was examined. The advantage of this system for the scope of the present thesis is its simplicity and predictability due to the symmetry of loads and geometry. Nevertheless, the implementation of a FE model allows complex structural systems to be solved as well, like:
  - Continuous beams where the highest stresses appear at multiple locations along the beams longitudinal axis
  - Constructions with holes or girder notches
  - Spatial structures

- The optimisation schemes used in this work are currently only capable of dealing with wooden boards of equal length and beams consisting of one lamella per beam layer. In order to further optimise the load-bearing behaviour an extension towards allowing multiple boards per layer, connected by finger joints, is conceivable. Herein the location of the finger joints could be subject to the optimisation as well, meaning the algorithm would be capable of cutting boards and removing knot groups to further improve the final structural system.
- Beside changes in the definition of the optimisation task, the used optimisation schemes could be enhanced as follows:
  - Combining metaheuristic algorithms. A common approach is to combine local search algorithms with genetic algorithms. Such combinations are referred to as memetic algorithms [36]. By combination of such algorithms it is possible to take advantage of the special capabilities of each algorithm. Like for memetic algorithms the task of maintaining the overall scope of the search space is subject to the genetic algorithm and the refinement of single solutions is subject to the local search.
  - Using advanced versions of genetic algorithms like steady-state genetic algorithms.
  - Implementing parallel genetic algorithms, e.g by using the so-called island model [50], to reduce the computation time.
- Improving the metamodel for better predictions of deflections from the FE model. Herein especially methods from the field of machine learning like
  - support vector machines and
  - artificial neural networks

  could be applied. Especially artificial neural networks using deep learning prove to be applicable to a variety of different machine learning tasks, including regression.

Bibliography

[1] D. W. Aha. “Editorial”. In: *Artificial Intelligence Review* 11.1-5 (1997), pp. 7–10.

[2] J. E. Baker. “Reducing bias and inefficiency in the selection algorithm”. In: *Proceedings of the second international conference on genetic algorithms*. 1987, pp. 14–21.

[3] L. Bianchi, M. Dorigo, L. M. Gambardella, and W. J. Gutjahr. “A survey on metaheuristics for stochastic combinatorial optimization”. In: *Natural Computing* 8.2 (2008), pp. 239–287.

[4] T. Blickle and L. Thiele. “A Comparison of Selection Schemes used in Evolutionary Algorithms”. In: *Evolutionary Computation*. Citeseer. 1997.

[5] C. Blum and A. Roli. “Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison”. In: *ACM Computing Surveys* 35.3 (2003), pp. 268–308.

[6] L. Bottou. “Large-scale machine learning with stochastic gradient descent”. In: *Proceedings of COMPSTAT’2010*. Springer, 2010, pp. 177–186.

[7] R. F. Coelho, M. Herrera, M. Xiao, and W. Zhang. “On-line Metamodel-Assisted Optimization with Mixed Variables”. In: *Evolutionary Algorithms and Metaheuristics in Civil Engineering and Construction Management*. Springer, 2015, pp. 1–15.

[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT Press, 1990.

[9] T. Cover and P. Hart. “Nearest neighbor pattern classification”. In: *IEEE transactions on information theory* 13.1 (1967), pp. 21–27.

[10] K. De Jong. “Parameter setting in EAs: a 30 year perspective”. In: *Parameter setting in evolutionary algorithms*. Springer, 2007, pp. 1–18.

[11] J. L. Devore. *Probability and Statistics for Engineering and the Sciences*. Cengage Learning, 2010.

[12] S. A. Dudani. “The distance-weighted k-nearest-neighbor rule”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 4 (1976), pp. 325–327.

[13] J. Ehlbeck and F. Colling. “Die Biegefestigkeit von Brettschichtholzträgern in Abhängigkeit von den Eigenschaften der Brettlamellen”. In: *Bauen mit Holz* 89.10 (1987), pp. 646–655.

[14] G. Fink, A. Frangi, and J. Kohler. “Bending tests on GLT beams having well-known local material properties”. In: *Mater Struct* 48.11 (Oct. 2014), pp. 3571–3584.

[15] R. W. Floyd. “Nondeterministic Algorithms”. In: *ACM* 14.4 (1967), pp. 636–644.

[16] C. Foley. “A three-dimensional paradigm of fiber orientation in timber”. In: *Wood Science and Technology* 35.5 (2001), pp. 453–465.

[17] C. Foley. “Modeling the effects of knots in structural timber”. Ph.D. Thesis. Division of Structural Engineering, Lund Unversity, 2003.

[18] B. Fox and M. McMahon. “Genetic operators for sequencing problems”. In: *Foundations of genetic algorithms* 1 (1990), pp. 284–300.

[19] M. Gendreau and J.-Y. Potvin. “Tabu Search”. In: *Handbook of metaheuristics*. Springer, 2010, pp. 41–59.

[20] M. Gendreau, J.-Y. Potvin, et al. *Handbook of Metaheuristics*. 2nd ed. International Series in Operations Research & Management Science 146. Springer US, 2010.

[21] D. E. Goldberg. *Genetic Algorithms in Search, Omptimization & Machine Learning*. Addison-Wesley Publishing Company, Jan. 1989.

[22] P. Guindos and M. Guaita. “A three-dimensional wood material model to simulate the behavior of wood with any type of knot at the macro-scale”. In: *Wood science and technology* 47.3 (2013), pp. 585–599.

[23] C. Hackspiel, K. de Borst, and M. Lukacevic. “A numerical simulation tool for wood grading model development”. In: *Wood science and technology* 48.3 (2014), pp. 633–649.

[24] K. Hofstetter, C. Hellmich, and J. Eberhardsteiner. “Development and experimental validation of a continuum micromechanics model for the elasticity of wood”. In: *European Journal of Mechanics-A/Solids* 24.6 (2005), pp. 1030–1053.

[25] J. H. Holland, C. Langton, and S. W. Wilson. *Adaption in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, 1992.

[26] C.-J. Johansson. “Grading of timber with respect to mechanical properties”. In: *Timber engineering* (2003), pp. 23–43.

[27] N. Jorge and J. W. Stephen. *Numerical optimization*. Springer, 2006.

[28] G. Kandler, J. Füssl, E. Serrano, and J. Eberhardsteiner. “Effective stiffness prediction of GLT beams based on stiffness distributions of individual lamellas”. In: *Wood Science and Technology* 49.6 (July 2015), pp. 1101–1121.

[29] G. Kandler, M. Lukacevic, and J. Füssl. “An algorithm for the geometric reconstruction of knots within timber boards based on fibre angle measurements”. In: *Construction and Building Materials* 124 (2016), pp. 945–960.

[30] D. E. Knuth. *The Art of Computer Programming Vol. 2*. Addison–Wesley, 1997.

[31] P. Larranaga, C. M. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic. “Genetic algorithms for the travelling salesman problem: A review of representations and operators”. In: *Artificial Intelligence Review* 13.2 (1999), pp. 129–170.

[32] M. López-Ibáñez, J. Dubois-Lacoste, L. P. Cáceres, M. Birattari, and T. Stützle. "The irace package: Iterated racing for automatic algorithm configuration". In: *Operations Research Perspectives* 3 (2016), pp. 43–58.

[33] H. R. Lourenço, O. C. Martin, and T. Stützle. "Iterated local search: Framework and applications". In: *Handbook of metaheuristics*. Springer, 2010, pp. 363–397.

[34] M. Lukacevic and J. Füssl. "Numerical simulation tool for wooden boards with a physically based approach to identify structural failure". In: *European Journal of Wood and Wood Products* 72.4 (2014), pp. 497–508.

[35] O. Maron and A. W. Moore. "The racing algorithm: Model selection for lazy learners". In: *Lazy learning*. Springer, 1997, pp. 193–225.

[36] P. Moscato and C. Cotta. "A modern introduction to memetic algorithms". In: *Handbook of metaheuristics*. Springer, 2010, pp. 141–183.

[37] A. Olsson and J. Oscarsson. "Three dimensional fibre orientation models for wood based on laser scanning utilizing the tracheid effect". In: *WCTE 2014, World Conference on Timber Engineering, Quebec City, Canada, August 10-14, 2014*. 2014.

[38] A. Olsson, J. Oscarsson, E. Serrano, B. Källsner, M. Johansson, and B. Enquist. "Prediction of timber bending strength and in-member cross-sectional stiffness variation on the basis of local wood fibre orientation". In: *European Journal of Wood and Wood Products* 71.3 (2013), pp. 319–333.

[39] *ÖNORM DIN 4074-1: Sortierung von Holz nach der Tragfähigkeit. Teil 1: Nadelschnittholz*. Norm. Sept. 1, 2012.

[40] *ÖNORM EN 14080: Holzbauwerke – Brettschichtholz und Balkenschichtholz – Anforderungen*. Norm. Aug. 1, 2013.

[41] *ÖNORM EN 1912: Bauholz für tragende Zwecke – Festigkeitsklassen – Zuordnung von visuellen Sortierklassen und Holzarten*. Norm. Oct. 15, 2013.

[42] J. Oscarsson, E. Serrano, A. Olsson, and B. Enquist. "Identification of weak sections in glulam beams using calculated stiffness profiles based on lamination surface scanning". In: *WCTE 2014, World Conference on Timber Engineering, Quebec City, Canada, August 10-14, 2014*. Université Laval. 2014.

[43] H. Petersson. "Use of optical and laser scanning techniques as tools for obtaining improved FE-input data for strength and shape stability analysis of wood and timber". In: *IV European conference on computational mechanics, Paris, France*. 2010.

[44] C. R. Reeves. "Genetic algorithms". In: *Handbook of metaheuristics*. Springer, 2010, pp. 109–139.

[45] C. R. Reeves. "Genetic algorithms for the operations researcher". In: *INFORMS journal on computing* 9.3 (1997), pp. 231–250.

[46] C. R. Reeves. "Using Genetic Algorithms with Small Populations." In: *ICGA*. Vol. 590. 1993, p. 92.

[47] E. Serrano and B. Enquist. *Mechwood II Glulam tests*. Tech. rep. Linnaeus University Växjö Sweden, 2014.

[48] S.-P. Simonaho, J. Palviainen, Y. Tolonen, and R. Silvennoinen. "Determination of wood grain direction from laser light scattering pattern". In: *Optics and Lasers in Engineering* 41.1 (2004), pp. 95–103.

[49] E.-G. Talbi. *Metaheuristics: from design to implementation*. Vol. 74. John Wiley & Sons, 2009.

[50] D. Whitley. "An overview of evolutionary algorithms: practical issues and common pitfalls". In: *Information and software technology* 43.14 (2001), pp. 817–831.