BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF CIVIL ENGINEERING

FAKULTA STAVEBNÍ

INSTITUTE OF STRUCTURAL MECHANICS

ÚSTAV STAVEBNÍ MECHANIKY

OPTIMIZATION OF PRESTRESSED CONCRETE TENDON PATH

OPTIMALIZACE TRASY PŘEDPÍNACÍCH KABELŮ

BACHELOR'S THESIS

BAKALÁŘSKÁ PRÁCE

AUTHOR | Monika Středulová

AUTOR PRÁCE

SUPERVISOR | doc. Ing. JAN ELIÁŠ, Ph.D.

VEDOUCÍ PRÁCE

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STAVEBNÍ

Studijní program	B3607 Stavební inženýrství
Typ studijního programu	Bakalářský studijní program s prezenční formou studia
Studijní obor	3647R013 Konstrukce a dopravní stavby
Pracoviště	Ústav stavební mechaniky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student	Monika Středulová
Název	Optimalizace trasy předpínacích kabelů
Vedoucí práce	doc. Ing. Jan Eliáš, Ph.D.
Datum zadání	30. 11. 2017
Datum odevzdání	25. 5. 2018

V Brně dne 30. 11. 2017

prof. Ing. Drahomír Novák, DrSc. Vedoucí ústavu	prof. Ing. Miroslav Bajer, CSc. Děkan Fakulty stavební VUT

PODKLADY A LITERATURA

Jaroslav Navrátil, Předpjaté betonové konstrukce, CERM, 2008, ISBN 978-80-7204-561-7
Kalyanmoy Deb, An efficient constraint handling method for genetic algorithms, Computer Methods in Applied Mechanics and Engineering, 186, 2-4, 2000, 311-338, ISSN 0045-7825
Marco Montemurro, Angela Vincenti, Paolo Vannucci, The Automatic Dynamic Penalisation method (ADP) for handling constraints with genetic algorithms, Computer Methods in Applied Mechanics and Engineering, 256, 2013, 70-87, ISSN 0045-7825

ZÁSADY PRO VYPRACOVÁNÍ

Student/studentka vytvoří optimalizační program založený na genetických algoritmech napojený na TDA modul, který umožní hledat optimální trasu předpínacícho kabelu pro různě zatížené konstrukce.

STRUKTURA BAKALÁŘSKÉ PRÁCE

VŠKP vypracujte a rozčleňte podle dále uvedené struktury:
1. Textová část VŠKP zpracovaná podle Směrnice rektora "Úprava, odevzdávání, zveřejňování a uchovávání vysokoškolských kvalifikačních prací" a Směrnice děkana "Úprava, odevzdávání, zveřejňování a uchovávání vysokoškolských kvalifikačních prací na FAST VUT" (povinná součást VŠKP).
2. Přílohy textové části VŠKP zpracované podle Směrnice rektora "Úprava, odevzdávání, zveřejňování a uchovávání vysokoškolských kvalifikačních prací" a Směrnice děkana "Úprava, odevzdávání, zveřejňování a uchovávání vysokoškolských kvalifikačních prací na FAST VUT" (nepovinná součást VŠKP v případě, že přílohy nejsou součástí textové části VŠKP, ale textovou část doplňují).

doc. Ing. Jan Eliáš, Ph.D.
Vedoucí bakalářské práce

ABSTRACT

The thesis explores possibilities of applying genetic algorithms on the problem of finding the optimal prestressed concrete tendon path. The objective of the thesis is to develop a genetic algorithm based on the Automatic Dynamic Penalization method and to test its robustness on selected analytical functions. Subsequently, the algorithm is connected to a Time Dependent Analysis module for the computation of prestressed concrete structure to solve selected examples of prestressed beams in the form of a constrained optimization problem. The algorithm is developed in the Python programming language with the help of the Distributed Evolutionary Algorithm library.

KEYWORDS

optimization, constraints, genetic algorithms, automatic dynamic penalization, prestressed concrete, tendon path

ABSTRAKT

Bakalářská práce se zabývá možnostmi využití genetických algoritmů pro optimalizaci trasy předpínacích kabelů. Cílem práce je vyvinout genetický algoritmus na základě metody automatické dynamické penalizace a ověřit jeho robustnost na vybraných analytických funkcích. Následně je tento algoritmus napojen na TDA modul pro výpočet předpětí a použit pro řešení optimalizační úlohy s omezením. Veškeré úlohy jsou řešeny v programovacím jazyce Python s využitím knihovny Distributed Evolutionary Algorithms.

KLÍČOVÁ SLOVA

optimalizace, omezení, genetické algoritmy, automatická dynamické penalizace, předpjatý beton, trasa předpínacích kabelů

STŘEDULOVÁ, Monika. *Optimization of prestressed concrete tendon path*. Brno, 2018, 47 p. Bachelor's Thesis. Brno University of Technology, Faculty of Civil Engineering, Department of structural mechanics. Advised by doc. Ing. Jan Eliáš, Ph.D.

ROZŠÍŘENÝ ABSTRAKT

Beton je známý svou malou pevností v tahu, která je přibližně desetkrát menší než pevnost v tlaku. Pro překonání tohoto nedostatku, se beton vyztužuje, a to buď ocelovou výztuží, čímž vznikne železobeton, nebo předpínací výztuží, čímž vznikne beton předpjatý. Postup předpínání, tedy vnesení tlakového napětí do betonové konstrukce, byl patentován na konci 80. let 18. století. Rozmach tohoto materiálu nastal až roku 1928, kdy Eugéne Freyssinet poprvé použil vysokopevnostní předpínací výztuž [10]. Navzdory počátečním neúspěchům způsobených velkými ztrátami vnesené předpínací síly, výzkum vlastností tohoto materiálu pokračoval a dnes jsou konstrukce z předpjatého betonu široce využívané. Navzdory dlouholetému výzkumu i širokému využití je stále velmi složité navrhnout optimální trasu předpínací výztuže. Jednoduchým postupem může být dosažen návrh splňující dané podmínky, ale pro navržení nejlepší, optimální trasy je nutné zapojit moderní výpočetní metody.

Proces hledání nejlepšího řešení jakéhokoliv problému se nazývá optimalizace. Problém k řešení se dá zpravidla definovat funkcí $f(\boldsymbol{x})$, která se označuje jako cílová. Ta je závislá na proměnných, které se sdružují do vektoru $\boldsymbol{x}$. Řešení může být případně omezeno dalšími podmínkami. Definičním oborem cílové funkce jsou potom možné hodnoty proměnných a optimálním řešením problému je potom globální extrém této funkce.

Metody řešení optimalizačních problémů se v zásadě dají rozdělit do dvou skupin: tradiční, které využívají analytické postupy, a moderní, které ve většině případů fungují na principech přírodních zákonů a pro řešení je ve většině případů nutné využít výpočetní kapacitu počítačů [12].

Genetický algoritmus, který spadá do metod moderních, je robustním nástrojem, který se pro optimalizaci trasy předpínacích kabelů velmi dobře hodí. Pracuje s množinou možných řešení, kterou v průběhu výpočtu obměňuje. Právě díky této souběžnosti výpočtu se hodí pro optimalizaci cílových funkcí, u kterých se předpokládá větší množství lokálních extrémů a větší množství proměnných, což je i problém optimalizace trasy předpínacích kabelů. Další výhodou je poměrně snadná implementace dalších podmínek, které musí řešení splňovat, taktéž relevantní pro problém řešený v této bakalářské práci.

Základním stavebním kamenem genetických algoritmů je využití evolučních principů přežití silnějších a dědičnosti. Algoritmus pracuje v cyklech a v každém upraví množinu (také „generaci“) možných řešení (také „jedinců“) pomocí genetických operátorů, které zákony evoluce a dědičnosti aplikují. Tímto způsobem algoritmus v každém cyklu nalézá lepší řešení.

Prvním genetickým operátorem je výběr jedinců ze současné generace. Tento krok vybere jedince, na které budou následně aplikovány další dva genetické operá-

tory, křížení a mutace. V rámci křížení si jedinci vymění část své informace mezi sebou. Pomocí tohoto kroku by mělo dojít k prohledání okolí již nalezených dobrých řešení. Mutace slouží opačnému účelu, a to k prohledání celého definičního oboru. Pomocí náhodné změny části řešení by měla zabránit předčasné konvergenci k lokálnímu optimu. Nicméně konkrétní forma nejen těchto genetických operátorů, ale každého kroku výpočtu, není jasně daná, protože záleží na potřebách řešeného problému [12]. Kromě toho jsou jednotlivé kroky algoritmu závislé na číselných parametrech, které můžou mít na robustnost algoritmu velký vliv.

Vyvinutý algoritmus byl tedy testován. První zkouškou byla ověřena robustnost hodnotící funkce zahrnující penalizaci řešení. Na jejím základě jsou vybíráni jedinci pro křížení a mutaci. Ohodnocení bylo implementováno podle [9]. V článku je navržená metoda hodnocení testována na analytické funkci, která je vystavena dvěma různým omezením, které nesplňují požadovaná omezení. Vyvinutý algoritmus byl testován na stejné funkci celkem desetkrát a průměrný výsledek z těchto spuštění algoritmu vykazoval stejně dobré výsledky jako referenční algoritmus uvedený v článku.

V druhé fázi byl analyzován vliv jednotlivých parametrů, které určují mají vliv na robustnost algoritmu. Cílem bylo nalezení nejefektivnějších hodnot. Byla zvolena základní sada parametrů a následně byla vždy jedna z hodnot změněna tak, aby bylo možné pozorovat její vliv na výpočet. Pro účely zkoušení byla optimalizována trasa předpínacího kabelu prostě podepřeného nosníku se spojitým zatížením ve směru osy y s konstantním průřezem. Pro účely optimalizace je trasa předpínací výztuže definovaná pomocí kvadratické Bézierovy křivky, která je jednoznačně určená pomocí tří bodů [1]. Cílovou funkcí je počet kabelů, kterou se snažíme z ekonomických důvodů minimalizovat. V tomto konkrétním případě vektor neznámých proměnných obsahoval jeden člen vyjadřující polohu trasy a druhý člen vyjadřující počet použitých kabelů.

Celkem sedm parametrů bylo postupně otestováno, na základě čehož byly následně stanoveny jejich optimální hodnoty a tak byl algoritmus nastaven pro budoucí optimalizace. Vyvinutý a otestovaný algoritmus byl následně aplikován na dvě konkrétní úlohy. Jednalo se o optimalizace trasy předpínacích kabelů prostě podepřeného nosníku a spojitého nosníku o dvou rozdílných polích. Délka prostého nosníku a prvního pole spojitého nosníku je stejná, přičemž i ostatní charakteristiky jako zatížení, charakteristické pevnosti betonu i oceli byly totožné pro oba případy.

Cílem bylo nalézt řešení s co nejmenším počtem kabelů, které vyhovuje třem následujícím podmínkám: v nosníku nevzniká tah od dlouhodobého zatížení, největší tlak je omezen a dráha kabelu musí být uvnitř nosníku. V prvním případě se jednalo o troj rozměrný problém, kde dvě neznámé vyjadřovaly tvar Bézierovi křivky a třetí počet kabelů. Druhý případ byl čtyř rozměrný, kde tři neznámé vyjadřovaly tvar dvou Bézierových křivek (jedna křivka pro každé pole) a poslední neznámá vyjadřovala počet kabelů.

Výpočet byl opět proveden desetkrát, přičemž v každém z celkem dvaceti výpočtů algoritmus nalezl řešení, které splňuje zadané podmínky. Z výsledků vyplývá, že nejvíce omezující podmínkou byla podmínka nulového tahu po délce konstrukce. Každé nalezené řešení bylo přesně na hranici tahu, tedy maximální tlak byl přesně 0 MPa. Nicméně u obou úloh se nalezené trasy podstatně lišily. To může být důsledkem přítomnosti několika lokálních minim s velmi podobnou hodnotou cílové funkce a tedy nedostatečnou robustností algoritmu.

DECLARATION

ACKNOWLEDGEMENT

I would like to express my gratitude to the supervisor of the thesis, doc. Ing. Jan Eliáš, Ph.D. for all the help and for all the valuable advice he has given me. His willingness and kindness never ceased to amaze me. I would also like to thank my better half, who has been an incredible support through my studies and who makes my life better every day. Last, but definitely not the least, I want to thank my parents for supporting me no matter what and always listening to me.

Brno . . . .

author's signature

Contents

1 Introduction **1**

2 Objectives of the thesis **2**

3 Engineering optimization **3**

3.1 The theory behind optimization 3

3.2 Optimization methods 5

4 Genetic algorithms **9**

4.1 The development of genetic algorithms 9

4.2 Basic principles 10

5 Prestressed concrete **15**

5.1 Basic design principle 15

5.2 Secondary effects 17

5.3 Losses of prestress 18

6 The optimization of a prestressed concrete tendon path **21**

6.1 Formulation of the problem 21

6.2 The developed algorithm 23

6.3 Analysis of the algorithm 26

6.4 Optimization of tendons 36

7 Conclusion **45**

Bibliography **46**

List of Figures

3.1 Minimum of $f(\boldsymbol{x})$ equals to the maximum of $-f(\boldsymbol{x})$. 3
3.2 Illustration of a possible constrained surface of optimization. 5
4.1 Flowchart representing basic structure of a genetic algorithm [8]. . . . 11
5.1 Stress strain diagram comparing different types of concrete [10]. . . . 16
5.2 Stress distribution in different parts of a prestressed simply supported beam. 17
5.3 Illustration of secondary effects of prestressing. 18
6.1 Quadratic Bézier curve [1]. 22
6.2 Representation of the objective function and the vector of design variables. 23
6.3 Representation of the test case 1 of two design variables and one constraint. 28
6.4 Representation of the test case 2 of two design variables and two constraints. 29
6.5 Representation of the test case of two design variables. 30
6.6 The domain of the two dimensional problem. 31
6.7 The performance of the algorithm for different N_{I}. 32
6.8 The performance of the algorithm for different number of individuals in the first generation. 32
6.9 The performance of the algorithm for different number of elite individuals N_{E}. 33
6.10 The performance of the algorithm for different crossover probability p_{c}. 33
6.11 The performance of the algorithm for different crowding degree of crossover η. 34
6.12 The performance of the algorithm for different mutation probability p_{m}. 35
6.13 The performance of the algorithm for different σ_{ini} and σ_{fin}. 35
6.14 Representation of the vector of design variables. 37
6.15 The convergence of each computation by the algorithm. 38
6.16 Solutions found by the algorithm. 38
6.17 Comparison of solutions found by the algorithm. 39
6.18 Normal stress of the best solution. 40
6.19 Representation of the vector of design variables. 41
6.20 The convergence of each run of the algorithm. 42
6.21 Solutions found by the algorithm. 43
6.22 Comparison of solutions found by the algorithm. 44
6.23 Normal stress of the best solution. 44

List of Tables

6.1 Comparison of results of the two algorithms for test case 1. 27
6.2 Comparison of results of the two algorithms for test case 2. 29
6.3 Results of the optimization of the simply supported beam's tendon path sorted from the best solution to the worst. 39
6.4 Results of the optimization of the continuous beam's tendon path sorted from the best solution to the worst. 43

1 Introduction

Without a doubt, the main advantage of prestressed concrete over the reinforced and plain ones is its load bearing capacity. By prestressing, additional compressive stress is induced in a concrete beam by high-strength steel tendons resulting in the ability to resist greater loads before cracking. Although known since the late 1880 when first prestressed concrete structures were patented, the boom in development began in 1928 when Eugéne Freyssinet used high-strength wires for prestressing. However, inventors at that time had to deal with great loss of prestressing due to shortening of concrete. In their earliest experiments, two thirds of induced stress disappeared [10].

These days, after ninety years of research, the loss of prestressing due to various aspects of the structure is well documented. But it doesn't necessarily make the design of a structure subjected to so many changes in time any less difficult. Various calculus-based approaches which had been put to practice appear to have one thing in common: finding only a passable solution. Furthermore, considerable experience might be crucial in the process.

The obvious answer is to use modern computation methods which became available due to fast computer development in recent years. Nonetheless, in order to find the optimal solution, the ideal tendon path, it could be convenient to use an advanced computational method - an algorithm specifically designed to find the minimum/maximum of noisy functions.

For such a task, a genetic algorithm was chosen. Genetic algorithms being methods based on the natural law of selection and heredity, and being population based as well, their popularity as problem solvers has grown considerably in recent years [9]. By mimicking nature and using its laws, genetic algorithms are robust optimization tools which are used not only in civil engineering [3]. Moreover, their structure allows casting additional constraints to possible solutions, which is highly relevant in the case of prestressed concrete structure design.

The purpose of the thesis is to develop a genetic algorithm, while implementing the evaluation as suggested by [9], to test the robustness of the algorithm, obtain the correct form of the algorithm by testing and finally apply the algorithm to solve the constraint optimization problem in the form optimization of a prestressed concrete tendon path. The result would be savings with regard to both the amount of material needed and the cost of the material.

2 Objectives of the thesis

Primary goals of the thesis are:

- to develop the genetic algorithm in Python programming language [11] based on the Automatic Dynamic Penalization method [9].
- to test the developed algorithm on selected analytical functions to asses its robustness.
- to determine the most effective forms of the algorithm by analysing its performance on a simple example of the optimization problem.
- to optimize selected examples of prestressed concrete tendon path by the developed algorithm.

3 Engineering optimization

The chapter aims to define general optimization problem and to give an overview of most common methods used in the field of engineering optimization. The term of optimization is introduced, followed by possible approaches to optimization along with their advantages and disadvantages. If not stated otherwise, following chapter is based on the book Engineering optimization: Theory and practice [13].

3.1 The theory behind optimization

Engineers from various fields can find themselves in an uneasy place as demands to their designs are often contradictory. One example for all from the field of civil engineering is the ever present effort to design a structure of a certain load bearing capacity for a minimum cost. But each of the decision making problem, including those of civil engineering, can be expressed as a function of varying number of variables $f(\boldsymbol{x})$. Such a function, called an objective function, expresses the nature of the given problem and summarizes criteria of the optimization, where the perfect (optimal) solution $\boldsymbol{x}_0$ is minimizing or maximizing such function. The process of finding the minimum/maximum value is called optimization. Further more, maximum of a function $f(\boldsymbol{x})$ equals to the minimum of the negative function $-f(\boldsymbol{x})$ and accordingly, minimum of a function equals to the maximum of the negative function, as displayed in Fig. 3.1. Consequently, each optimization problem can be defined as a minimization of a function.

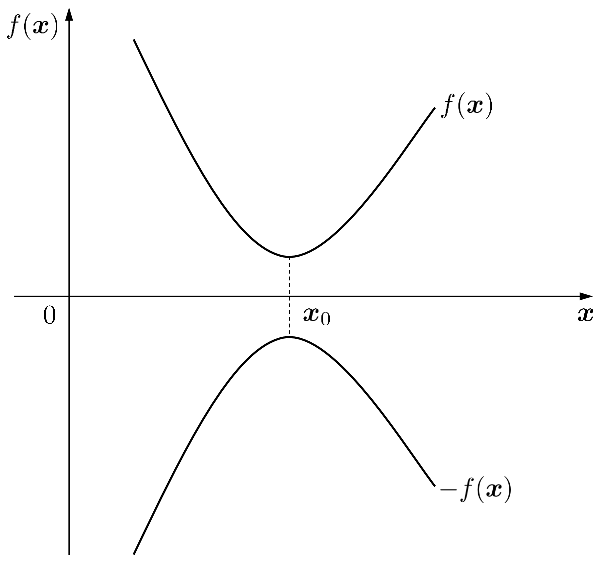


Fig. 3.1: Minimum of $f(\boldsymbol{x})$ equals to the maximum of $-f(\boldsymbol{x})$.

There are two major optimization categories, based on the nature of the problem at hand. First category are problems defined only by the objective function $f(\boldsymbol{x})$ and the unknowns of vector $\boldsymbol{x}$. Such a task is called unconstrained optimization. In the second category, additional conditions are cast on the solution in the form of equality and inequality constraints, hence the name constrained optimization. See the definition of a optimization problem in subsequent equations.

$$\text{find } \boldsymbol{x}_0 \text{ so that for all } \boldsymbol{x} : f(\boldsymbol{x}_0) \leq f(\boldsymbol{x}) \tag{3.1}$$

$$\boldsymbol{x} = \{x_1, x_2, x_3, \ldots, x_{\mathrm{n}}\} \tag{3.2}$$

$$g_{\mathrm{j}}(\boldsymbol{x}_0) \leq 0;\ j \ = \ 1, 2, 3, \ldots, m \tag{3.3}$$

$$h_{\mathrm{k}}(\boldsymbol{x}_0) = 0;\ k \ = \ 1, 2, 3, \ldots, n \tag{3.4}$$

The function $f(\boldsymbol{x})$ is the objective function to minimize and $\boldsymbol{x}$ is the multidimensional vector of n design variables. In the definition of constrained problem, g_{j} are known inequality constraints and h_{k} are known equality constraints to which the solution is subjected. The definition domain of any optimization is defined as the domain of the vector of design variables:

$$\boldsymbol{x}_{\mathrm{L}} \leq \boldsymbol{x} \leq \boldsymbol{x}_{\mathrm{U}} \tag{3.5}$$

where $\boldsymbol{x}_{\mathrm{L}}$ and $\boldsymbol{x}_{\mathrm{U}}$ are lower and upper bounds of the vector of design variables respectively. Based on above mentioned equations, a hypothetical two-dimensional design space may be drawn for illustration as in Fig 3.2. In the picture, feasible region is bounded by four constraints: g_1, g_2, g_3 and g_4. Four types of points can be identified based on their position in relation to constraints:

- free acceptable point: positioned in the feasible region,
- bound acceptable point: placed on the boundary of feasibility,
- free unacceptable point: located in the non-feasible region but not lying on any constraint surface,
- bound unacceptable point: positioned on the boundary of one or more constraints but being non-feasible due to other constraint(s).

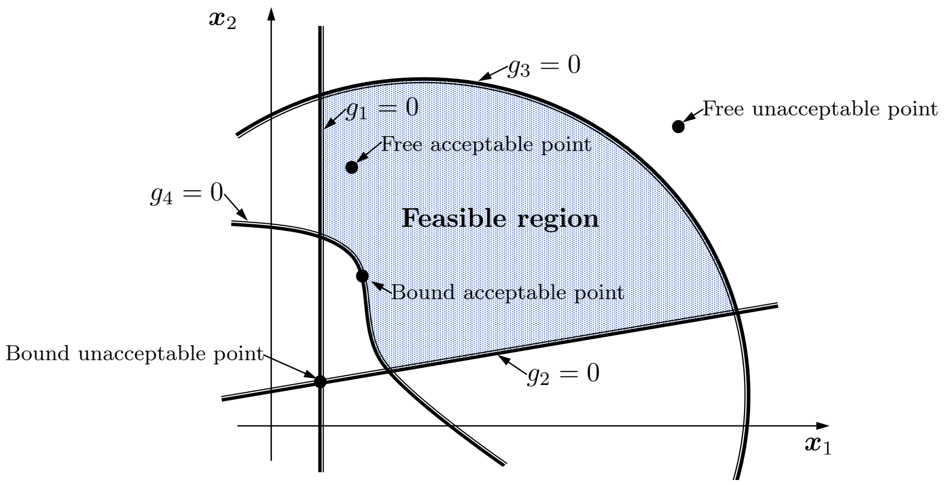


Fig. 3.2: Illustration of a possible constrained surface of optimization.

3.2 Optimization methods

Based on the historical development and the development of means of optimization, various methods used for engineering optimization can be sorted into two main categories: traditional or classic ones and modern techniques. In the following sections, each category will be described along with examples.

3.2.1 Traditional techniques

Although varying in the approach, roots of traditional search methods can be traced back to the times of Newton or Leibnitz. These methods were then further developed in the twentieth century and implemented with the help of digital computers. However, due to the fact that real-life problems are often represented by noisy objective functions with multiple local minima and maxima and the search space of such functions might be considerably wide, traditional methods do not always posses the robustness required for solving complicated, real-life cases. Despite that, these methods can prove highly efficient when applied in combination with modern search methods, as described in section 3.2.2. Following paragraphs offer basic description of each of the three main types of traditional search methods [3].

Calculus-based methods

Based on analytical approach, traditional methods may be divided into two main subclasses based on the procedure: direct and indirect.

The first one, direct method, uses local gradient. By obtaining gradient at a given point, it is then possible to move in the direction of the steepest slope. Such an approach is also called a hill-climbing method. However, when dealing with an objective function of multiple local minima/maxima, climbing based on the gradient is highly inefficient, as it searches only in a local region. Further more, if local extrema are present, the success of this method is highly dependable on the starting point.

The second type of calculus-based methods are indirect methods. When searching for the extremum, set of equations is obtained by setting the gradient to zero. By solving these equations, possible extrema are found, which are then tested to achieve the global maximum or minimum.

Both approaches, whether direct or indirect, are local in scope and dependable on the existence of derivative. Thus, objective functions has to be differentiable and continuous. Real-life problem representations often do not posses such characteristics and thus make calculus-based methods suited only to a limited problem domain [3].

Enumerative methods

When using enumerative method, the search algorithm examines one value in the search space after another to find the extreme one. The domain has to be either finite or discretized. Although attractive in its simplicity, such an approach lacks the efficiency when applied to larger search spaces. As such, its application to practical problems is highly restricted.

Random search algorithms

As the name suggests, random search examines random values of the search space and saves the best one found so far. However, just like enumerative methods, its apparent lack of robustness makes the method highly unsuitable for larger scale problems.

3.2.2 Modern Techniques

Along with the development of modern computers, not only traditional methods had been improved, but also new methods had begun to emerge in the second half of twentieth century. Most of the methods mimic principles found in nature, whether hereditary principles or behaviour of colonies of organisms. Five methods will be introduced in the following sections.

Simulated Annealing

The simulated annealing simulates the slow cooling process of metal to achieve the global minimum. The algorithm works with one solution, gradually moving in the search space. By contrast to gradient based methods, a temperature-like parameter T is introduced to control the search process. Also, a probability distribution is implemented to generate the new value. In the beginning of the search, the value of T is usually higher and as the search continues, the value declines. The greater the parameter T, the greater the probability of accepting a worse solution. Towards the end of the search the improved solutions are accepted almost exclusively, thus searching in the local area.

Particle Swarm Optimization

Particle swarm optimization (PSO) mimics the behaviour of animal colonies (swarms), where the term 'particle' refers to one animal of such a colony. The swarm contains specified number of particles, each of which is characterized by its velocity and its position. Each particle moves in the search space, remembers the best solution it has found so far and tries to find a better one. The particles communicate between themselves, thus in the case of finding exceptional point, other particles will follow to further explore the region. The process of updating the swarm is repeated specified number of times.

Ant Colony Optimization

Based on the behaviour of ant colonies, the Ant Colony Optimization (ACO) mimics the approach ants use to find the shortest way to food. The optimization problem is defined as multi-layered where each layer corresponds to one design variable and each of the variables is defined by its permitted discrete values. Thus this approach is applicable only to discrete optimization problems, also called combinatorial problems. The ants move from one variable to the other leaving certain trace behind themselves (also called pheromone trace), the better the solution, the stronger the trace is. Ants which subsequently visits the solutions are then moving in the direction of stronger pheromone and are thus capable of finding the better solutions.

Optimization of Fuzzy Systems

On the contrary to previously mentioned methods, Optimization of Fuzzy Systems deals with real life problems which are stated in vague linguistic terms. With that being said, for the search of the optimum value of such systems, the basic definition of optimization changes. In contrary to the definition (3.1), constraints are given in a fuzzy statements called membership functions, which are boundaries of the area where a statement is true. Optimum solution can be viewed as an intersection of such functions. The solution can be found only after membership functions are formulated, which requires substantial knowledge of the problem to optimize.

Genetic Algorithms

Genetic algorithms (GA) mimic two basic evolutionary principles found in nature, which are survival of the fittest along with hereditary principles. Through the computation a set of possible solutions is improved by applying the natural principles in the form of genetic operators. Due to the mixture of these principles, GAs are a very effective tool when searching in a domain of noisy function. However, a lot of testing is needed as there are parameters which need to be tuned specifically for a given problem.

3.2.3 Algorithm suitable for the optimization of prestressed tendon path

The above mentioned list is by no means complete, however, it represents the most common methods. The algorithm used for the optimization of prestressed concrete tendon path should reflect the demands of the problem.

The objective function of the problem is continuous on the specified domain, however, the presence of multiple local extrema can be presumed. Based on [13] an algorithm working with multiple possible solutions at the same time would be more effective with regard to the given problem.

From the requirements listed in the previous paragraph, traditional optimization techniques can be rejected as their robustness appears not be sufficient for a given problem. Regarding modern optimization methods, the Ant Colony Optimization is suited for discrete domains only and the Optimization of Fuzzy Systems is not fitting the clearly defined problem such as the optimization in question. Both Genetic Algorithms and Particle Swarm Optimization appear to be well suited for the problem, as both work with the set of solutions, thus being supposedly robust for multiple local optima problems. For the purpose of the thesis the optimization via Genetic Algorithms had been chosen.

4 Genetic algorithms

The chapter firstly defines a genetic algorithm based on the historical development of the optimization method. Subsequently, basic structure of a GA is described with its key steps. The most common ways of implementation of each step are outlined along with possible alternatives.

4.1 The development of genetic algorithms

For the reasons described in previous chapter a genetic algorithm are suitable tool for the purpose of the optimization. They are stochastic [1] algorithms [13] which combine the law of natural selection along with the principles of genetics to become an algorithm robust enough to be applicable to a wide range of optimization problems [3].

First efforts to mimic natural processes via computer simulations had appeared in the nineteen sixties and their primary goal was to study principles of evolution. As evolution is a process which takes long periods of time to be observable in nature, computer simulations allowed examining development of these processes more easily. Such practices were later described by John Holland in his book "Adaptation in natural and artificial systems" (1975) with the purpose to study adaptability of natural systems and possibilities to apply these principles in algorithms. As Mitchell remarks in [8], the theoretical background of genetic algorithms as described by Holland had laid foundations to their future development. Basic principle of Holland's algorithm was to replace the existing set of possible solutions to a given problem by a new one by applying principles of natural selection and heredity. Possible solutions were coded in a binary representation (strings of zeros and ones) and hereditary principles were applied in a form of binary genetic operators: crossover and mutation [5].

In the following years, genetic algorithms started to attract more attention as a possible optimization tool. However, their limitations started to present themselves as well. Application of GA upon various problems had proved that in many cases binary representation is not sufficient and better results are accessible by so-called real-coding. Instead of binary representation, real-coded algorithms use different representation of solutions, better corresponding to a given problem: matrices, arrays, vectors etc. Also, binary genetic operators were altered to better express nature of the problem. These transformed forms of GA were given various names: non-standart GA, modified GA or specialized GA [7].

[1] probabilistic

Genetic algorithms are not the only algorithm inspired by the process of natural selection. All such algorithms are members of a group called evolutionary algorithms. However, because of the gradual development of genetic algorithms, the difference between real-coded, modefied GAs and other evolutionary algorithms is not possible to define [7], [12]. For the purpose of this work, the expression *genetic algorithm* will be interchangeable with *transformed genetic algorithm* and *evolutionary algorithm.*

4.2 Basic principles

The basic structure of a genetic algorithm is displayed in Fig. 4.1 on the following page. This structure is in no way complete as it can be enriched by other steps. The deployment of these steps as well as their specific form is determined based on the given problem, as genetic algorithms tend to be more efficient the more specialized they are [7]. The meaning of each step is explained in subsequent paragraphs. The implementation of each step and their modifications are explained primarily with regard to the optimization solved in later chapters of the thesis, which is the optimization of a constrained problem by a real-coded algorithm.

Because of the inspiration by nature, the terminology corresponds to the one of biological evolution and heredity. Previously mentioned set of possible solutions is also called a "generation" or a "population", while a solution itself is referred to as an "individual". As seen on the flowchart (Fig 4.1), the computation is iterative and a new generation of individuals is generated in each step. Thus number of iterations (cycles) corresponds to the number of generations N_{G} used in one computation by the algorithm.

4.2.1 Selection of the first generation

To begin the computation, the first generation comprising N_{I} has to be created. The most common way is to generate the required number of individuals randomly out of given definition domain. However, the first step of the search may be facilitated by certain precautions, which mainly depend on the knowledge of the problem at hand. Naturally, the wider the domain is, the more solutions are there to explore. Hence the possibilities are to either generate more individuals in the first generation (first suggested by De Jong in [6]) or to narrow the domain by applying knowledge about the problem [12]. The ultimate goal is to found better solutions in the first generation which would guide the search in the later stages of computation. As suggested by Simon in [12], the initial population may be also locally optimized

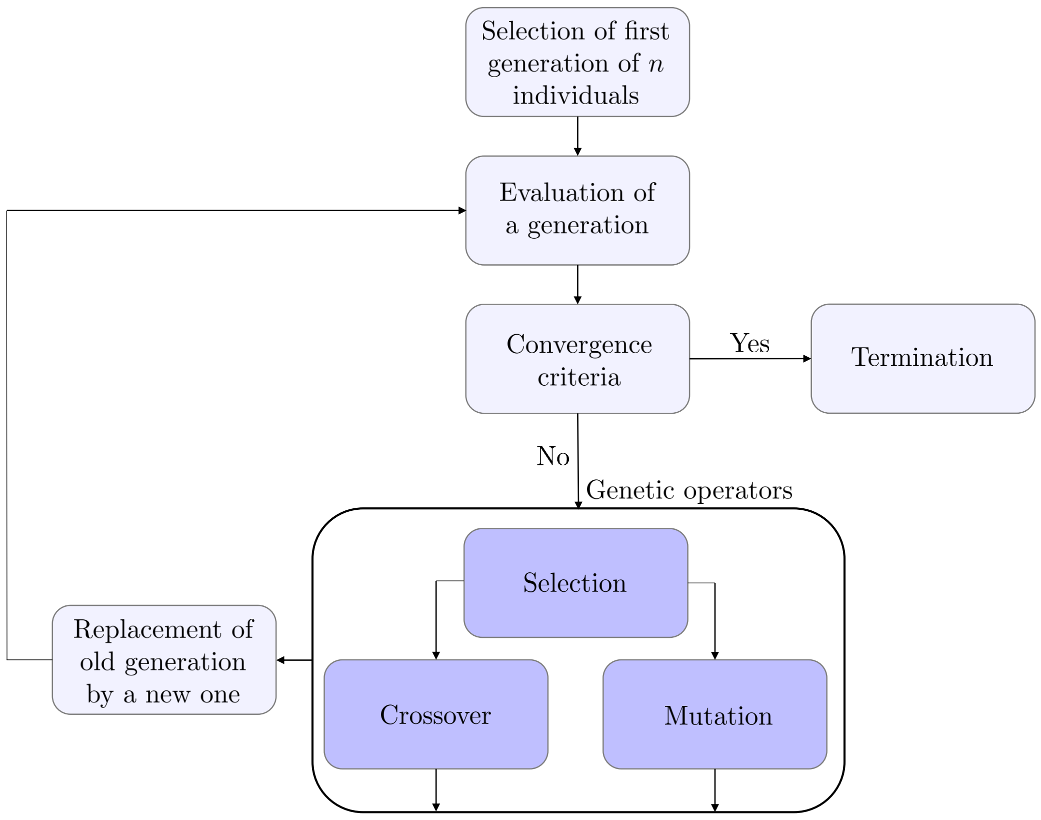

Selection of first generation of n individuals
Evaluation of a generation
Convergence criteria
Yes
Termination
No
Genetic operators
Selection
Crossover
Mutation
Replacement of old generation by a new one

Fig. 4.1: Flowchart representing basic structure of a genetic algorithm [8].

to obtain even better first generation of solutions. However, such a modification requires the use of an additional algorithm.

4.2.2 Evaluation of the generation

After selecting the first generation, each of the individuals is evaluated. Such an evaluation is also referred to as a fitness score of the individual. In the case of a constrained problem, the evaluation is calculated based on two components.

Fitness function

Firstly, the realization of the objective of the optimization is assessed. From the definition of an optimization (Eq. 3.1) can be derived, that the smaller the value of the objective function is for the given individual, the better evaluation it receives. The fitness function may mathematically vary from the objective function, however, both functions pursuit the same target: to express the problem [12].

Penalization function

Second part of the evaluation represents the compliance with the additional conditions, expressed in the form of inequality and equality constraints, (3.3) and (3.4) respectively. Their fulfilment is evaluated by a penalization function. Many practical optimization problems place the optimum value on the boundary of feasible and infeasible regions. For this reason, it is desirable to use such a penalty, which favours feasible individuals only to a certain extent so that best non-feasible individuals can still act as attraction points when looking for a better solution.

The formulation of such a function may vary considerably, based on the problem at hand and multiple approaches exist when dealing with penalization [9]. As the penalization has to be scaled appropriately with regard to the values of the fitness function to fulfil the above mentioned objective, it is necessary to use a coefficient which adjusts the penalty. To avoid manual tuning of the factor which requires multiple testing, various methods have been suggested to balance between feasible and non-feasible regions during computation. In these methods, the penalty function is self-adaptive and the coefficient varies in generations (see [9] for an overview of self-adaptive methods). In the developed algorithm the Automatic Dynamic Penalization method was implemented as described in the section 6.2.2.

4.2.3 Convergence criteria and termination

As the algorithm runs in cycles, a convergence criteria must be specified to terminate the computation. The nature of the criteria depends on the problem at hand and can be in principle divided into one of the following three groups [12].

1 **Searching for a previously known solution.**
In this case, it is possible to stop the algorithm when the best solution in a generation lies close enough to a given one. As genetic algorithms are stochastic, the solution will never be absolutely precise and it is thus necessary to specify the degree of precision required of the algorithm.

2 **Preset number of cycles/evaluations.**
Specifying the number of generations is used in problems for which the solution is unknown. As the number of generations corresponds to the number of cycles, the algorithm will terminate after given number of iterations is performed. The approach has its advantages, such as run-time predictability and simplicity. However, when comparing different algorithms, the sole number of generations N_{G} used is not a valid criteria as the number of individuals N_{I} may vary. For this reason, a total number of evaluations performed N_{TOT} is preferred.

3 **No improvement criteria.**
Common philosophy behind these types of termination condition is following: if the fitness value of individuals over generations is no longer improving, the algorithm has found the optimal value. Obvious drawback is that the algorithm may be terminated before converging to the global optimum. The condition may be specified in various terms, whether the fitness of the best individual in a generation, the average fitness of all individuals or a standard deviation of fitness in a generation.

4 **Combination of the above mentioned.**
A combination of the above mentioned criteria may be used as well. For example to limit the number of cycles while using the no improvement criteria as well.

4.2.4 Genetic operators

Genetic operators apply the principles of natural selection and heredity. Through appropriately implementing them, two opposite objectives are pursued. Properly setted genetic operators should balance the exploitation of previously found solutions while exploring the whole definition domain. If an emphasis is given to exploitation, the algorithm may converge prematurely towards local optima and even though, the exploitation of such local area will be sufficient, the final solution will not satisfy the goal of the optimization. In the opposite case of accenting exploration, the region found will be that of the global optimum, however, the result will not be precise enough.

Genetic operators are usually applied in two stages. In the first step, individuals are selected from a generation(principle of the survival of the fittest) while in the second step, the selected individuals are combined or changed to produce the individuals of the new generation (principles of heredity).

Selection

To incorporate the survival of the fittest, the selection should be performed at least partly on the basis of evaluation. However, an emphasis too strong on the evaluation might lead to premature convergence towards local optimum. Oppositely, a completely random selection might result in not exploiting the present information. Multiple stochastic methods of selection have been designed. The most common is perhaps the roulette-wheel selection, when the better fitness the individual has, the better chance of being selected [12].

A frequent tool integrated in selection is a so-called elitism. First suggested by De Jong in [6], elitism sends a number of good enough individuals directly to the new generation. Such a tool aims at preventing the algorithm from changing the best solutions found so far in the generation by subsequent genetic operators.

Crossover

Crossover primarily aims to exploit the already found solutions contained in the current generation. By swapping parts of individuals between themselves, new individuals are generated.

In its simplest form, crossover may be performed as one-point recombination, as described by the following equations [12]. Vectors $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ are two individuals selected from the current generation, containing k number of variables.

$$\begin{aligned} \boldsymbol{x}_a &= \left\{x_1^a, x_2^a, ..., x_{m-1}^a, x_m^a, ..., x_{k-1}^a, x_k^a\right\} \\ \boldsymbol{x}_b &= \left\{x_1^b, x_2^b, ..., x_{m-1}^b, x_m^b, ..., x_{k-1}^b, x_k^b\right\} \end{aligned} \tag{4.1}$$

while vectors $\boldsymbol{y}_a$ and $\boldsymbol{y}_b$ represent their offsprings, members of the new generation. Where n is the number of design variables in every individual and the crossover point is right before the mth component. By recombining $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ individuals via one-point crossover, following offsprings are produced:

$$\begin{aligned} \boldsymbol{y}_a &= \left\{x_1^a, x_2^a, ..., x_{m-1}^a, x_m^b, ..., x_{k-1}^b, x_k^b\right\} \\ \boldsymbol{y}_b &= \left\{x_1^b, x_2^b, ..., x_{m-1}^b, x_m^a, ..., x_{k-1}^a, x_k^a\right\} \end{aligned} \tag{4.2}$$

Just like in the case of previous steps, multiple alternations may be made. One point crossover is generally considered the least productive one [12]. In lieu of it a multiple-point crossover may be used, or other forms such as multi-parent crossover or binary like crossover in the case of real-coded algorithm. For other options see [7, 12].

The chance that each of the selected individuals will be recombined is expressed by the crossover probability p_c. It ranges from 0 when no individual will be chosen, to 1 when every individual will be recombined.

Mutation

In opposition to crossover, mutation primarily serves the exploration of the domain and thus the preservation of a diversity in a generation. By randomly changing parts of a solution, a completely different solution should be found, potentially better but at least different then those already found. The aim is to avoid premature convergence towards local optimum.

The mutation probability p_m defines the probability that each variable in every selected individual will be mutated. The new variable is added the current one is usually generated either from a uniform distribution or from a Gaussian distribution. Both of these two options are defined by the mean and by the standard deviation and may considerably influence the search of the algorithm.

5 Prestressed concrete

As the objective of the optimization is to determine the optimal prestressed concrete tendon path, following chapter explains basic principle of prestressed concrete structures. Comparison of a plain concrete and a reinforced concrete is drawn. Subsequently, secondary effects along with the losses of prestressing are described as both may have considerable effects on the distribution of internal forces of a prestressed concrete structure and thus play an important role in the optimization process. If not stated otherwise, the chapter is based on the book [10] by Jaroslav Navrátil.

5.1 Basic design principle

Well known characteristics of a plain concrete is its low strength in tension. The tensile strength f_{ctm} is approximately ten times smaller compared to the compressive one f_{ck}. The response of such a structure loaded in tension is approximately linear until first crack appears, resulting in brittle failure happening without any previous warning.

This serious drawback of any concrete structure can be outweighed by inserting reinforcement in the form of steel rods. Such a reinforcement is placed in the most critical parts of the structure, where tension might occur. The reinforcement then passively resists the load applied. As the reinforcement is of a ductile material, sudden failure is avoided when the stress reaches the yield strenght of the material. Even though the stiffness of the structure is reduced, its ability to resist loading remains.

To prevent the reduction of stiffness, the creation of cracks and subsequently corrosion of the reinforcement, prestressed concrete had been developed. By prestressed tendons an additional compressive stress is induced in the structure resulting in further delaying the formation of cracks and thus the stiffness reduction. Further more, there is a possibility to control the distribution of internal forces by the prestressing force and by the tendon path.

In following stress-strain diagram the response of different kinds of concrete structures to loading is shown. Black colour represents plain concrete, a reinforced concrete is represented by a blue one and a prestressed concrete by a red one. The hatched part corresponds to the improvement of a prestressed concrete over reinforced one as the area bellow the curve is proportional to the energy necessary for a failure of the structure. The picture is taken from [10] and theoretically compares the same elements subjected to the same loading, which is not possible in practice.

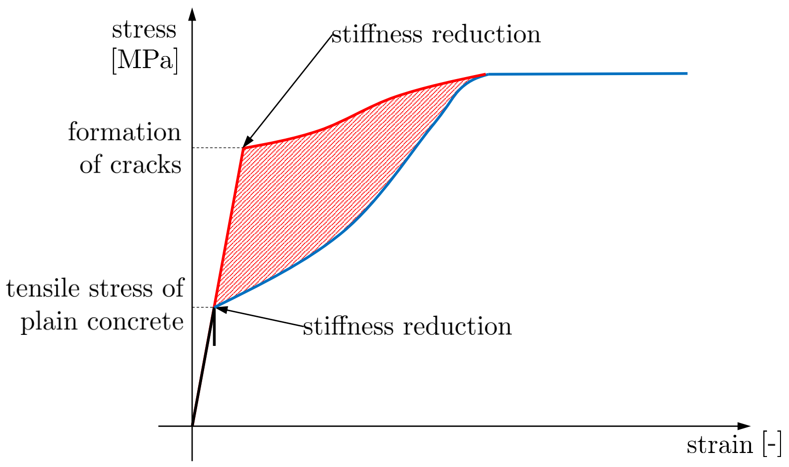


Fig. 5.1: Stress strain diagram comparing different types of concrete [10].

The mechanism of eliminating normal stress from loading by prestressing is described in following equations:

$$\sigma_{\mathrm{c}} = \sigma_{\mathrm{c,P}}^{\mathrm{N}} + \sigma_{\mathrm{c,P}}^{\mathrm{M}} + \sigma_{\mathrm{c,l}}^{\mathrm{M}} \ \left(+\sigma_{\mathrm{c,l}}^{\mathrm{N}}\right) \tag{5.1}$$

$$\sigma_{\mathrm{c}} = \frac{N_{\mathrm{P}}}{A} + \frac{M_{\mathrm{P}} z}{I_y} + \frac{M_{\mathrm{l}} z}{I_y} + \left(\frac{N_{\mathrm{l}}}{A}\right) \tag{5.2}$$

In the first equation σ_{c} represents the resulting stress in the beam, the first letter of the upper index stands for the internal force (M for a bending moment and N for normal force) and the second letter stands for the origin of the force (P for prestressing and l for external loading and self-weight). In the following formula, I_{y} stands for the moment of inertia of the cross section of the beam in the direction of y axis and z indicates the distance from the centre of the cross section. In the case of applying only horizontal external loading, no normal stress $\sigma_{\mathrm{c,l}}^{\mathrm{N}}$ emerges (as in example in Fig. 5.2). However, in a case of structures loaded differently, it influences the stress in the construction as well.

Tension is considered of positive value and compression of a negative value, thus when the additional compression is induced by prestressing, considerable amount of the occuring tensile stress may be eliminated. See the graphical representation of the stress midspan and at the end of a simply supported beam subjected to continuous load g and prestressing force N_{P} in Fig. 5.2.

The diagrams show that to eliminate the tensile stress effectively, the tendon should induced the biggest compression in the area which is most likely to be subjected to tension: the place of the maximum bending moment from external loading and self-weight. In this case that is the middle part of bottom fibres. However, as the external bending moment decreases towards the supports, different stress occurs at the end of the beam, see bottom part of the Fig. 5.2. The position of the tendon path in the lower part of the cross section may lead to a high negative bending moment in the top fibres as there is no bending moment from external loading present.

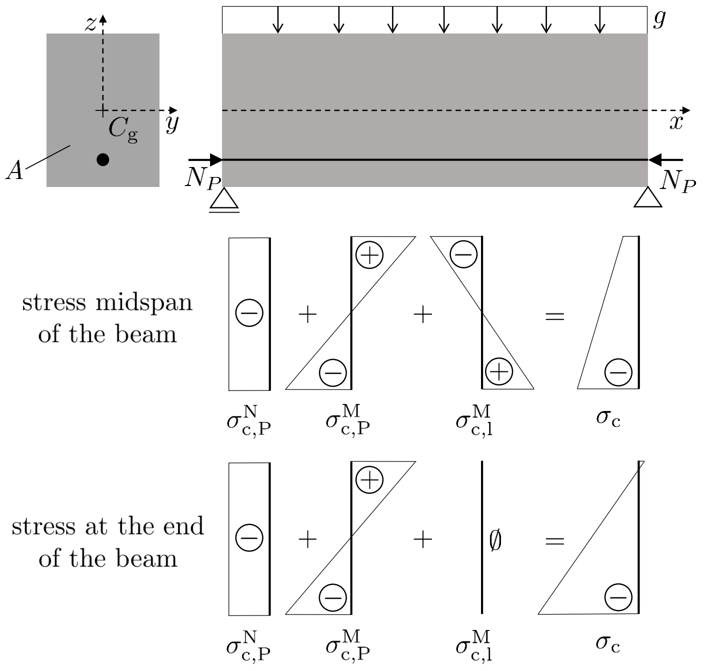


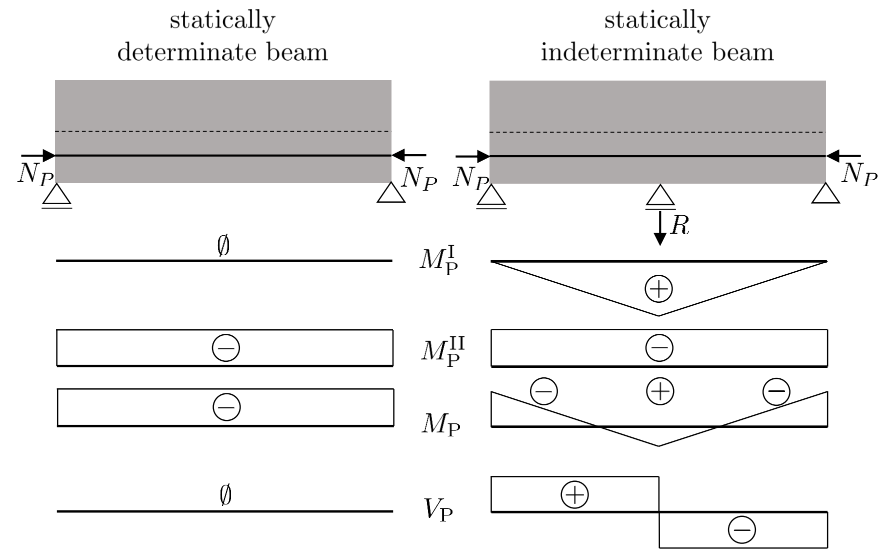
Fig. 5.2: Stress distribution in different parts of a prestressed simply supported beam.

5.2 Secondary effects

In the Fig. 5.2, the stress in a simply supported prestressed beam is demonstrated. However, statically indeterminate structures are subjected to additional stress due to statically indeterminate forces. These effects are called complementary or secondary effects of prestressing. The term "primary prestressing effects" describes the statically determinate reactions of prestressing, also termed basic effects. The total effects of prestressing can be then calculated as a summation of primary (I) and secondary (II) effects. The total moment of prestressing can be written as follows:

$$M_{\mathrm{P}} = M^{\mathrm{I}}_{\mathrm{P}} + M^{\mathrm{II}}_{\mathrm{P}} \tag{5.3}$$

The comparison of internal forces of statically determinate and indeterminate beams is shown in Fig. 5.3. An additional reaction R occurs due to the pressence of the middle support of statically indeterminate beam. The support restrains deformation of the beam, thus additional forces arise: secondary bending moment M_P^II and secondary shear force V_P^II which equals to the resulting shear force V_P.



statically determinate beam
statically indeterminate beam
N_P
N_P
N_P
N_P
R
$\emptyset$
M_P^I
$\oplus$
$\ominus$
M_P^II
$\ominus$
$\ominus$
M_P
$\ominus$ $\oplus$ $\ominus$
$\emptyset$
V_P
$\oplus$
$\ominus$

Fig. 5.3: Illustration of secondary effects of prestressing.

5.3 Losses of prestress

Prestress as induced in the beam at the time of creation (prestressing) does not remains constant. Due to various factors which will be explained later in the section, a reduction occurs either by immediate action or in time. Such a reduction is also called loss of prestress and it equals to the difference between the initial prestressing value and the effective prestress remaining in a structure.

As indicated above, the losses may be divided into two groups from the perspective of time. The first type of losses occurs at the time of prestressing, and as such is termed immediate loss or short-term loss. The second type, occurring during service life of the structure, is called time dependent loss or long-term loss.

5.3.1 Short-term losses

In comparison to long-term losses, immediate losses are highly dependable of the technology used, whether the beam is pre-tensioned[1] or post-tensioned[2]. The following list classifies the losses with regard to the type of production.

[1]The reinforcement is prestressed first followed by a casting of the concrete.

[2]The concrete element is cast first and prestressed after hardening of the concrete.

- **Elastic shortening:** occurs with both type of manufacturing, although when post-tensioning only in case of seqential tensioning.
- **Friction:** although present in pre-tensioned structures as well (at the draping points of the reinforcement), this loss influences post-tensioned structures considerably more.
- **Anchorage slip:** only in case of post-tensioning as there is no need for anchor in the other procedure.

For the purpose of short term losses calculation, both concrete and tendon are considered ideally elastic. Also, the prestressing force is assumed to act in the tendon section's center of gravity.

When the force is induced in a pre-tensioned beam, its effect results in immediate deformation of the concrete beam which shortens. Simultaneously the tendon is also deformed, resulting in its shortening and thus, loss of prestress. If every cable in post-tensioned beam would be stressed at the same time (which is not feasible) no loss would occur due to elastic shortening. However, in most cases beams are prestressed one after another and due to the sequentiality the loss occurs as well. Due to the presumed bond between concrete and tendon, the strain increment in the concrete $\Delta\varepsilon_\mathrm{c}$ is equal to the strain reduction in the tendon $\Delta\varepsilon_\mathrm{p}$, see following equations.

$$\Delta\varepsilon_\mathrm{c} = \Delta\varepsilon_\mathrm{P} \tag{5.4}$$

$$\Delta\varepsilon_\mathrm{c} = \frac{\Delta N_\mathrm{c}}{A_\mathrm{c}E_\mathrm{c}} \tag{5.5}$$

$$\Delta\varepsilon_\mathrm{P} = \frac{P - \Delta N_\mathrm{c}}{A_\mathrm{P}E_\mathrm{P}} \tag{5.6}$$

ΔN_c is the increment of normal force in concrete which equals the reduction of prestressing force $\Delta N_\mathrm{c} = P - \Delta P$, P is the prestressing force and AE is the axial stiffness of both materials (marked by respective lower index).

Loss of prestress as a result of friction appears mainly in post-tensioned structures, as the friction exists between the tendon and walls of a tendon duct. In case of a pre-tensioned structure the loss appears at draping points of the reinforcements and between the tendon and the stressing bed. The total change of prestress between points A and B od distance l can be obtained as in following equation:

$$\int_{P_\mathrm{A}}^{P_\mathrm{B}} \frac{dP}{P} = -\mu \int_{\alpha_\mathrm{A}}^{\alpha_\mathrm{B}} d\alpha - K \int_{x_\mathrm{A}}^{x_\mathrm{B}} dl \tag{5.7}$$

where the total loss is a combination of the loss due to angular change α and along the tendon's length between the points l. P_A and P_B is the prestressing force applied in respective points, μ and K are the friction coefficients.

The loss due to anchorage slip always depends on the anchor used and the lenght of the slip w is reported by the manufacturer. The anchorage slip must be equal to the total shortening of the tendon $\varepsilon_{\mathrm{pw}}$ due to the drop of the stress during anchorage set $-\Delta\sigma_{\mathrm{pw}}$:

$$w = -\int_0^{l_{\mathrm{w}}} \Delta\varepsilon_{\mathrm{pw}} dl = -\int_0^{l_{\mathrm{w}}} \Delta\sigma_{\mathrm{pw}} dl \tag{5.8}$$

where the integral limit is not known [10].

5.3.2 Long-term losses

Time dependent losses occur after anchoring or transport and influence the value of prestressing force through the life span of the element. Unlike short-time losses they are not dependent on the production used. See major causes of long-term losses in the list below.

- **Relaxation of the tendon(s).**
- **Shrinkage and creep of concrete.**

In the case of tendons relaxation, an increased deformation over time occurs due to a constant stress. The deformation (elongation) increases and thus the tendon relaxes. Various factors influence the loss, namely the level of induced prestressing force and time. Shrinkage of concrete is the result of water evaporation from the concrete which occurs through the life span of the structure. As such, it does not depend on the loading history. The creep is caused by sustained load applied on the element.

The calculation of the problem is complex as the above mentioned phenomena depend on various factors, such as the concrete mix parameters, the design strength and the age when loading begins etc. Mostly numerical methods are used for the calculation, as the ageing is an obstacle to analytical solution.

However, the total strain in the element subjected to normal stress $\varepsilon_m athrmN$ may be formulated as follows [10]:

$$\varepsilon_m athrmN(x,t) = \sigma_{\mathrm{N}}(x,t_0)J(t,t_0) + \int_{t_0}^{t} J(t,t')\dot{\sigma}_{\mathrm{N}}(x,t')dt' \tag{5.9}$$

where the strain depends on two variables, x the position of stress and t the time. First part of the equation represents the instantaneous strain which is elastic if the stress is small [14]. t_0 is the time of applying the stress and $J(t,t_0)$ is the compliance function, also called the creep function. The function represents the strain at time t from the load applied at time t_0. The second part of the equation represents the inelastic, stress-dependent strain, where $\dot{\sigma}_{\mathrm{N}}(x,t')dt'$ represents the stress increment from time t_0 to time t.

6 The optimization of a prestressed concrete tendon path

To optimize prestressed concrete tendon path, firstly the parametrization of the tendon path is described to define the problem later. Subsequently, the algorithm itself is described along with the form of individual steps. One of the crucial steps has been the implementation of the penalty function so its performance on selected analytical function has been analysed. As the efficiency of the algorithm depends on certain constant values, the testing has been performed on a two-dimensional optimization example to asses the best values of these parameters. Results of the performed tests are presented. Last but not the least, the developed algorithm is applied on selected examples of the optimization problem and its results are described.

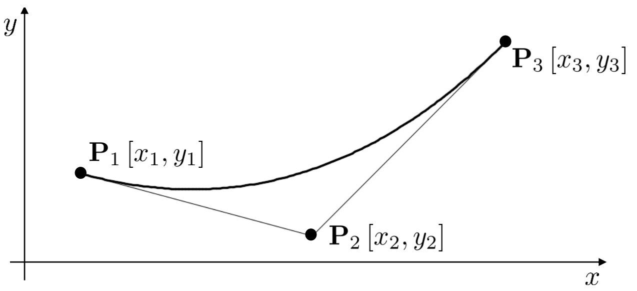
6.1 Formulation of the problem

The ultimate goal of the optimization is to design the tendon path in a way so that the least possible cables could be used while complying with given conditions. To do so, the algorithm is connected to Time Dependent Analysis module kindly provided by the supervisor of the thesis, doc. Ing. Jan Eliáš, Ph.D. The modul takes into account both short-term and long-term losses as described in section 5.3. With regard to the goal of the optimization, apart from the path of the tendon and the number of cables, all the other values needed for the optimization are selected prior to the computation (e.g. characteristic strengths of both materials, dimensions of the beam, loading, life span etc.). With regard to the behaviour of the concrete subjected to stress (Fig. 5.1), following conditions are cast on the optimal solution member:

1. no tension occurs in the structure due to the long-term loading,
2. maximum compression doesn't exceed 60% of the characteristic compressive strength of the concrete,
3. the tendon path is located inside the beam.

6.1.1 Parametrization of the tendon path

In order to successfully apply the developed optimization algorithm, the path of the cable has to be parametrized. For this purpose one quadratic curve is considered per span (or per beam, in case of a simply supported beam) representing the central line of the tendon. The connection of curves in case of a continuous beam is obtained by inserting an arch generated by the module. To parametrize the quadratic curve, a Bézier curve is used. The curve is in its quadratic form represented by three control points, each of which is defined by its x and y coordinate.



y
$\mathbf{P}_3\,[x_3, y_3]$
$\mathbf{P}_1\,[x_1, y_1]$
$\mathbf{P}_2\,[x_2, y_2]$
x

Fig. 6.1: Quadratic Bézier curve [1].

It is possible to calculate any point of the curve with the help of the parameter t, which is assigned a value of $t \in \langle 0, 1\rangle$. For example if $t = 0$ then $[x(0), y(0)]$ is the first control point of the curve and if $t = 0.5$ then $[x(0.5), y(0.5)]$ is the point in the middle of the curve. Each point of the curve may be calculated with the help of the parameter t by the following equation:

$$\mathbf{B}(t) = (1-t)^2 \mathbf{P}_1 + 2(1-t)\, t\mathbf{P}_2 + t^2\mathbf{P}_3 \tag{6.1}$$

Thus for the purpose of the optimization, the position of control points is subjected to optimization as they present unambiguous definition of the tendon's path.

6.1.2 Definition of the problem

Following equations define the given problem, which is termed as a constrained optimization due to the occurrence of additional conditions.

Objective function $f(\boldsymbol{x})$ equals to the number of cables, which is to be minimized:

$$\text{find } \boldsymbol{x}_0 \text{ so that for all } \boldsymbol{x} : f(\boldsymbol{x}_0) \leq f(\boldsymbol{x}) \tag{6.2}$$

$$f(\boldsymbol{x}) = n_{\text{cab}} \tag{6.3}$$

The n dimensional vector of unknowns is chosen with the least possible number of variables to allow the optimization without jeopardizing the process unnecessarily. For that reason x coordinates are fixed. As the path is defined by one Bézier curve per span of the beam, in case of a continuous beam more curves is used and only y of its control points are left free to be optimized. The last member of the vector is the number of the cables n_{cab}:

$$\boldsymbol{x} = \{y_1, y_2, \dots, y_{\text{n}}, n_{\text{cab}}\} \tag{6.4}$$

The domain is restricted by lower $\boldsymbol{x}_\mathrm{L}$ and upper $\boldsymbol{x}_\mathrm{U}$ bounds of the vector:

$$\boldsymbol{x}_\mathrm{L} \leq \boldsymbol{x} \leq \boldsymbol{x}_\mathrm{U} \tag{6.5}$$

The sign convention is used so stress in tension acquires positive value while compression is described by negative value. The first two of the following equation concern minimum and maximum stress occurring in the beam, while the final condition assures that the path is led inside the element by restricting extreme points of the curve, see Fig. 6.2.

$$g_1 : \sigma_\mathrm{max} \leq 0 \tag{6.6}$$

$$g_2 : \sigma_\mathrm{min} \geq 0.6 f_\mathrm{ck} \tag{6.7}$$

$$g_3 : \max\left\{|y(t)|\right\} \leq \frac{h}{2} \tag{6.8}$$

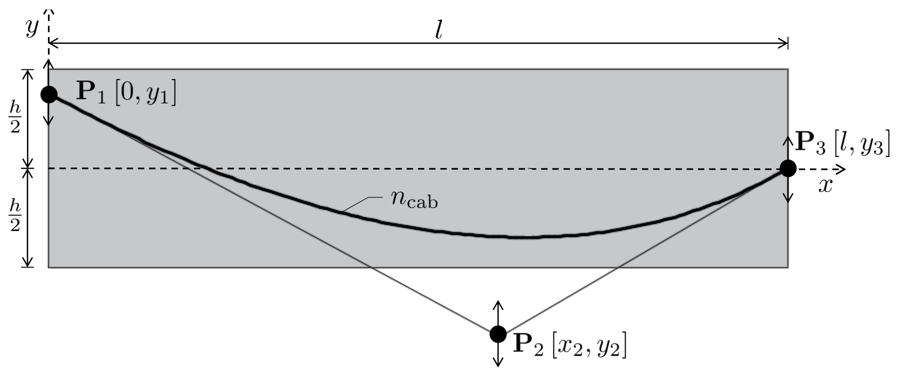


Fig. 6.2: Representation of the objective function and the vector of design variables.

6.2 The developed algorithm

The algorithm is written in the Python programming language [11] with the aid of its built in libraries and the Distributed Evolutionary Algorithms library [2]. The section describes the selected implementation of individual steps which were discussed in the section 4.2.

6.2.1 Selection of the first generation

The first generation is selected randomly with a uniform distribution. However, greater number of individuals, kN_I, is generated so the chance of finding a feasible individual in the first generation increases. Such procedure was first suggested by De Jong in [6]. The value of the multiplier k was subjected to testing and its influence on the performance of the algorithm is described later in section 6.3.

6.2.2 Evaluation of generations

An exterior penalty based method called The Automatic Dynamic Penalization method (ADP) was implemented as suggested in [9] with certain changes.

The purpose of the method is to allow the algorithm to search on the boundaries of non-feasibility [9]. The resulting evaluation of a non-feasible individual with respect to each condition is the same as the evaluation of the best individual in the generation. Such an approach is highly relevant in the case of the tendon path optimization, as the best results are expected to lie on the boundary of infeasible region. To enable such a search, non-feasible points are expected not to be eliminated by the evaluation and should remain in the population to act as attraction points.

The penalty function of ADP $f_{\mathrm{P}}(\boldsymbol{x})$ is defined as a modified objective function $f(\boldsymbol{x})$ [9]:

$$\text{find } \boldsymbol{x}_0 \text{ so that for all } \boldsymbol{x} : f_{\mathrm{P}}(\boldsymbol{x}_0) \leq f_{\mathrm{P}}(\boldsymbol{x}) \tag{6.9}$$

$$f_{\mathrm{P}}(\boldsymbol{x}) = f(\boldsymbol{x}) + \sum_{i=1}^{m} c_i \max\{0, g_i(\boldsymbol{x})\} + \sum_{i=1}^{n} q_j \max\{0, |h_j(\boldsymbol{x}) - \epsilon|\} \tag{6.10}$$

The first member of the equation is the objective function, the second member incorporates inequality constraints. The third one incorporates equality constraints by introducing acceptable tolerance ϵ, thus converting the equality constraint to the form of inequality constraint. For the purpose of the optimization in this whole thesis,only inequality constraints are considered. It can be noticed that when the conditions are not violated ($g_i(\boldsymbol{x}) \leq 0$), the penalization function equals to the objective function. In the case of any constraint violation, the evaluation is based on the degree of violation of a given condition g_i. The scaling coefficient of a constraint violation c_i is determined for each generation separately based on the individuals that violate i-th constraint $\boldsymbol{x}^{\mathrm{NF}_i}$.

$$c_i = \max_{\boldsymbol{x} \in x^{\mathrm{NF}_i}} \frac{\left| f(\boldsymbol{x}^{\mathrm{F}}_{\mathrm{BEST}}) - f(\boldsymbol{x}) \right|}{g_i(\boldsymbol{x})} \tag{6.11}$$

Where $\boldsymbol{x}^{\mathrm{F}}_{\mathrm{BEST}}$ is the best feasible individual in the generation. If no feasible individual appears in the generation, its value $f(\boldsymbol{x}^{\mathrm{F}}_{\mathrm{BEST}})$ is equal to zero.

6.2.3 Termination criteria

The simplest form of termination criteria has been chosen: the number of generations N_{G} determined prior to the computation. For the purpose of analysing the penalization function and the determination of coefficients, N_{G} was set to 200 generations. However, for the optimization of the tendon path, the number of generations is augmented to 500 generations in order to allow the algorithm to find a more precise solution. For the purpose of future comparisons, the total number of evaluations is given for each problem.

6.2.4 Selection of individuals for a new generation

The selection of individuals to undergo genetic operators differs for the first generation and the subsequent generations. In the case of the first generation, the number of individuals is multiplied by k. The selection is based solely on the evaluation and N_I individuals are chosen to undergo crossover and mutation and subsequently continue to second generation.

Since the second generation, the number of individuals remains constant (N_I) and the individuals are always selected based on the following key. Elitism is applied with regard to each constraint and to feasible individuals. m constraints are given, thus the number of elite individuals equals to $(m+1)N_\mathrm{E}$. The remaining $N_\mathrm{I} - (m+1)N_\mathrm{E}$ is chosen randomly. The value of N_E was subjected to testing (see section 6.2.2).

However, elite individuals are still present in the set from which the algorithm chooses so beside being directly chosen, they may be subjected to genetic operators as well. In such case both the elite form and the modified form of the individual continues to the next generation.

6.2.5 Crossover

A simulation of binary crossover is implemented. The crossover modifies in-place the input individuals element by element [2] for each of the k variables contained in an individual. The crossover is defined in the following equation [12]:

$$\begin{aligned} y_k^a =& 0.5\left[(1-\beta_k)\,x_k^a + (1+\beta_k)\,x_k^b\right] \\ y_k^b =& 0.5\left[(1+\beta_k)\,x_k^a + (1-\beta_k)\,x_k^b\right] \end{aligned} \tag{6.12}$$

β_k is computed for each individual based on a generated random number $r \in \langle 0, 1 \rangle$ [2]:

$$\beta_k = \begin{cases} r^{\frac{1}{\eta+1}}, & \text{if } r \le 0.5 \\ \left(\frac{1}{2(1-r)}\right)^{\frac{1}{\eta+1}} & \text{otherwise} \end{cases} \tag{6.13}$$

Where η stands for the crowding degree of the crossover, which expresses how much offsprings y_a and y_b resemble their parents x_a and x_b. The greater the value of η is, the more the offsprings resemble their parents. Although no bounds are given for η, the value is recommended between 0 and 5 [12]. For the purpose of the optimization, various values have been tested to obtain the best one for the problem at hand, see section 6.3.

6.2.6 Mutation

The mutation is defined as Gaussian, centered in the middle of the search domain. The value of the probability p_m (see section 4.2.4) was determined based on the testing presented in section 6.3. Standard deviation σ linearly decreases over generations, so that the exploitation is gradually more focused at regions already found. The starting value of standard deviation σ_{ini} was subjected to testing as well as the rate of decreasing of the value.

6.3 Analysis of the algorithm

The analysis of the algorithm was performed in two phases. Firstly, the algorithm was tested on selected analytical function with different constraints to asses robustness of the penalization function and analyse its implementation. Subsequently, the algorithm was applied to a two dimensional optimization problem of the tendon path to determine the most efficient values of the coefficients described in the previous section.

6.3.1 Test function

For the purpose of evaluating the performance of implemented penalization function, the function as defined in [9] was selected. As certain form of the penalization function is tested on the same function in the article, it is possible to draw a comparison of results presented in the article with results obtained by the developed algorithm.

The only difference between the algorithm developed and the reference one is the selection method. On the contrary to the algorithm described in [9], the developed algorithm uses elitism and random selection for the rest of individuals as explained in 6.2.4. However, the value of coefficients is set to be the same, so the number of generations $N_{\text{G}} = 100$ and number of individuals in one generation $N_{\text{I}} = 200$ making the total number of evaluations $N_{\text{TOT}} = 20000$. The number of individuals in the first generation is also equal to N_{I}. Probabilities of crossover and mutation equal to $p_{\text{c}} = 0.85$ and $p_{\text{m}} = \frac{1}{N_{\text{I}}} = 0.005$. Other steps are implemented as described in the section 6.2.4.

The function is of two variables, continuous on a given domain:

$$\text{find } \boldsymbol{x}_0 \text{ so that for all } \boldsymbol{x} : f(\boldsymbol{x}_0) \leq f(\boldsymbol{x})$$

$$f(x_1 x_2) = -\exp(0.2\sqrt{x_1^2 + x_2^2}) \sin(x_1) \cos(1.2 x_2) \tag{6.14}$$

$$\begin{aligned} 0 \leq x_1 \leq 4\pi \\ 0 \leq x_1 \leq 2\pi \end{aligned} \tag{6.15}$$

Test case 1

In the first case, the function (6.15) is subjected to the following constraint:

$$g_1(x_1, x_2) = \exp(0.012x_1^2) - 1 - x_2 \leq 0 \tag{6.16}$$

There is multiple local minima present on the feasible side of the domain and the constrained minimum is placed exactly on the boundary of infeasibility. Further more, the unconstrained minimum is placed closely to the unfeasible one.

Ten runs of the algorithm were performed. The following table presents mean and standard deviation of values obtained by the algorithm in comparison to the reference solution presented in [9]. The distance from the compared value was calculated from the mean value.

	x_1	x_2	$f(x_1, x_2)$
Mean	10.70716	2.95853	-8.11192
Standard deviation	0.01289	0.01311	0.00406
Reference solution [9]	10.71119	2.96646	-8.09933
Distance to the ref. solution			0.00889

Tab. 6.1: Comparison of results of the two algorithms for test case 1.

Tab. 6.1 documents, that the developed algorithm has found a solution better by 0.0889. Thus the algorithm is robust enough for the problem. The slightly changed penalization function combined with elitism with regard to non-feasible individuals does not negatively effects the algorithm.

The search domain along with the constraint g_1 can be seen in the Fig. 6.3. The solution lies on the boundary where $g_1 = 0$.

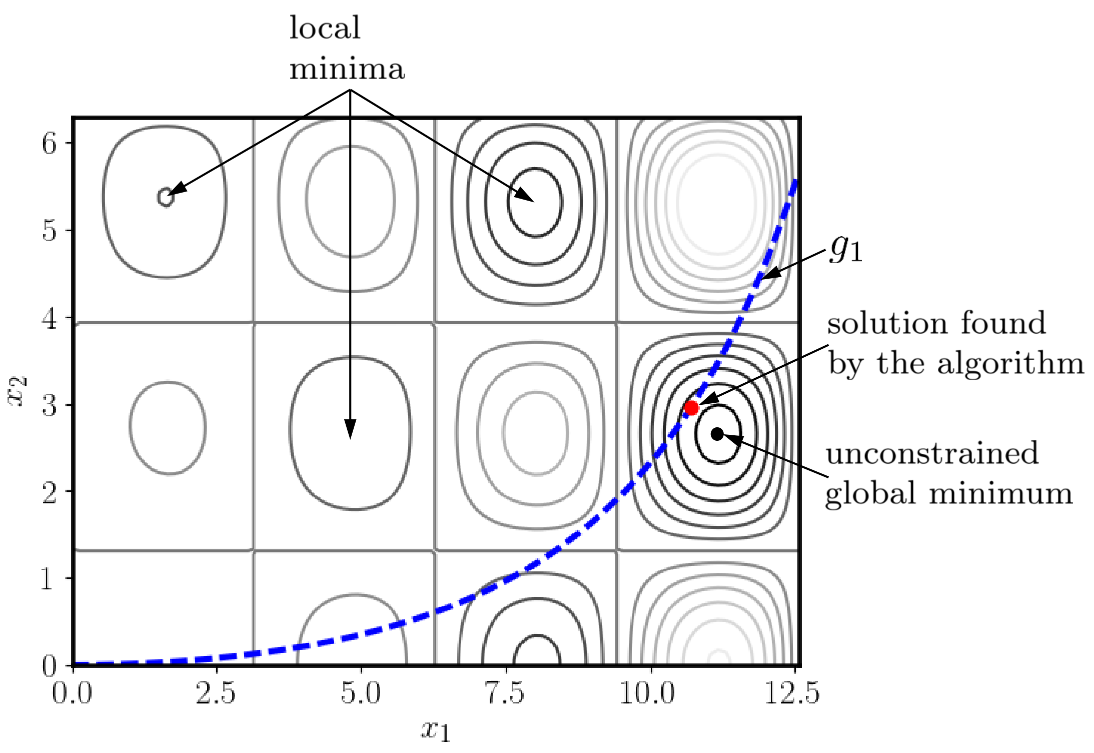


Fig. 6.3: Representation of the test case 1 of two design variables and one constraint.

Test case 2

For the second test the function is subjected to two constraints. There are several regions containing local minima, which are barely infeasible. Just like in the case of the first function, the unconstrained global minimum lies in the infeasible area. The constrained minimum lies on the intersection of the two constraints. The first condition is the constraint g_1 from test case 1, the second constraint g_2 is defined as follows:

$$g_1(x_1, x_2) = \exp^{0.012x_1^2} - 1 - x_2 \leq 0 \tag{6.17}$$

$$g_2(x_1, x_2) = 0.1\left[\left(\frac{0.4}{4\pi}x_1 + 0.7\right)\sin(\frac{\pi}{2}x_1)\sin(\frac{\pi}{2}x_2) - 0.7\right] \leq 0 \tag{6.17}$$

Ten runs were performed as in the previous case. Means and standard deviations are shown in the Tab. 6.2, along with results of the compared algorithm of [9]. The distance from the reference algorithm was calculated based on the mean value of $\boldsymbol{x}_0$.

The definition domain is depicted in Fig. 6.4 where barely infeasible islands are present as a result of the constraint g_2 (represented by white). Despite the fact that the global minimum lies on the intersection of the two constraints cast, the algorithm has again found the global minimum in each of the ten runs.

	x_1	x_2	$f(x_1, x_2)$
Mean	10.47196	2.72999	-7.47438
Standard deviation	0.00174	0.00230	0.01022
Reference solution [9]	10.45320	2.71465	-7.33770
Distance to the ref. solution			0.13800

Tab. 6.2: Comparison of results of the two algorithms for test case 2.

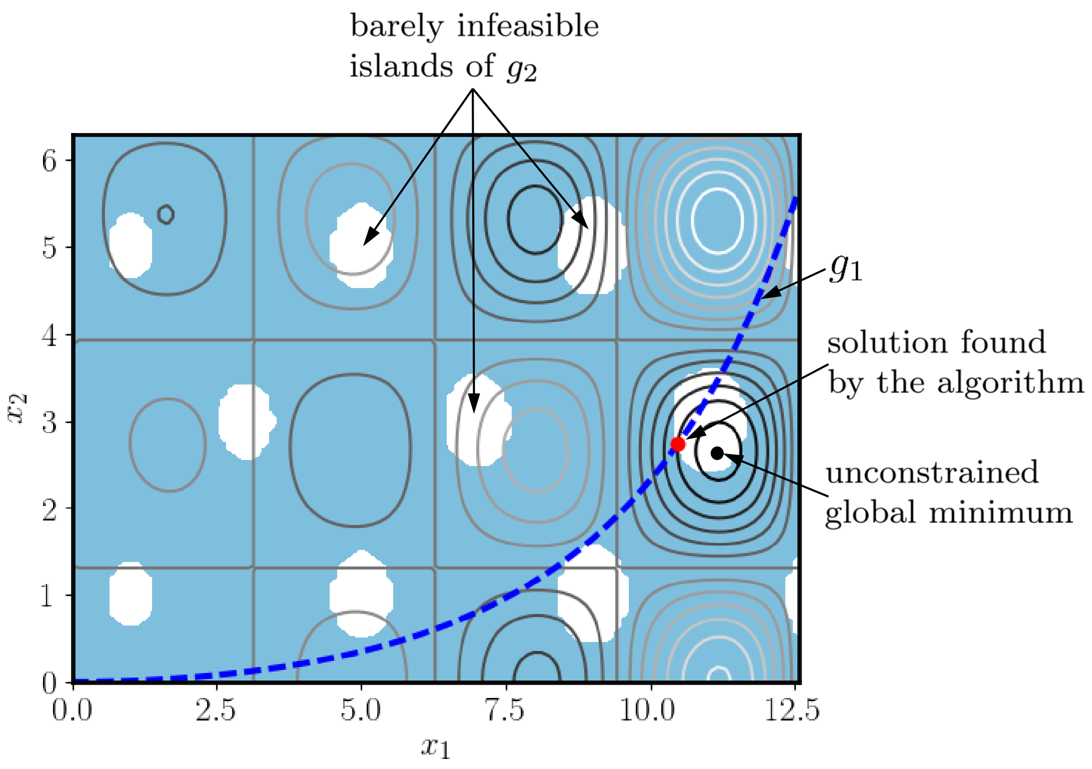


Fig. 6.4: Representation of the test case 2 of two design variables and two constraints.

6.3.2 Analysis of the effects of algorithm coefficients

To design an algorithm tailored to the given problem, determining of the optimal values of the coefficients have been performed. The values to determine were namely: the number of individuals in a population N_I, the number of individuals in the first generation (the multiplier k of N_I), the number of elite individuals transferred directly to a new generation N_E, the probability of crossover p_c, the crowding degree of crossover η, the probability of mutation p_m, and the change of standard deviation through generations. The testing was performed on the first version of the algorithm, where the evaluation was precisely as suggested in [9] and the elite individuals were chosen only with respect to their feasibility.

To obtain information about the algorithm's behaviour, basic set of the concerned values was given and only single parameter was tested at time. The basic set of values was set as follows: $N_\mathrm{I} = 100$, $k = 5$, $N_\mathrm{E} = 5$, $p_\mathrm{c} = 0.5$, $\eta = 5$ and $\sigma_\mathrm{ini} = 0.2$ and $\sigma_\mathrm{end} = 0.02$. For each test, ten runs of the algorithm were performed, subsequently its convergence was compared as well as the results. The information obtained is therefore limited, as the mutual interaction between different parameters is not taken into account. However, such analysis was not yet done due to its complexity and computational resources required.

The analysis was performed on a two dimensional problem of designing prestressed concrete tendon path of a simply supported beam. The curve is determined by one Bézier curve where the y-coordinate of the middle point y_2 is optimized. The other two control points of the curve are fixed in both directions. Second variable is the required number of cables n_cab. The three constraints as described in the section 6.1.2 are applied. Dimensions of the beam as well as loading along with times of application are given. The problem is defined by the following objective function, vector of design variables and domain:

$$f(\boldsymbol{x}) = n_\mathrm{cab} \tag{6.18}$$

$$\boldsymbol{x} = \{y_2, n_\mathrm{cab}\} \tag{6.19}$$

$$-3 \leq y_2 \leq 3 \tag{6.20}$$

$$20 \leq n_\mathrm{cab} \leq 40 \tag{6.21}$$

See the representation of the problem in Fig. 6.5.

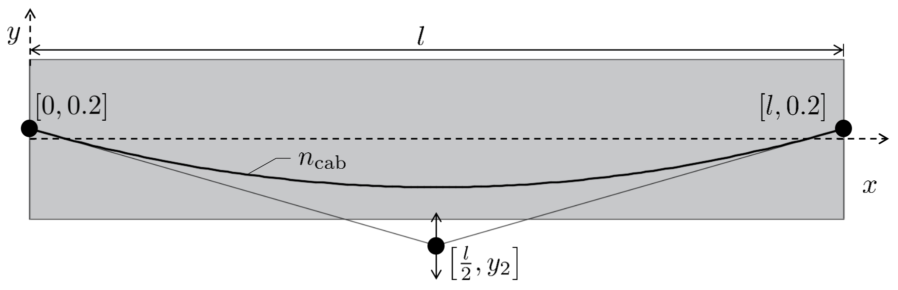


Fig. 6.5: Representation of the test case of two design variables.

The definition domain looks as shown in Fig. 6.6. Each colour represents feasible area with respect to one condition and constrained minimum is located. The first condition (6.6) concerning maximum stress is the most limiting one as its feasibility region represents the feasible region of the problem.

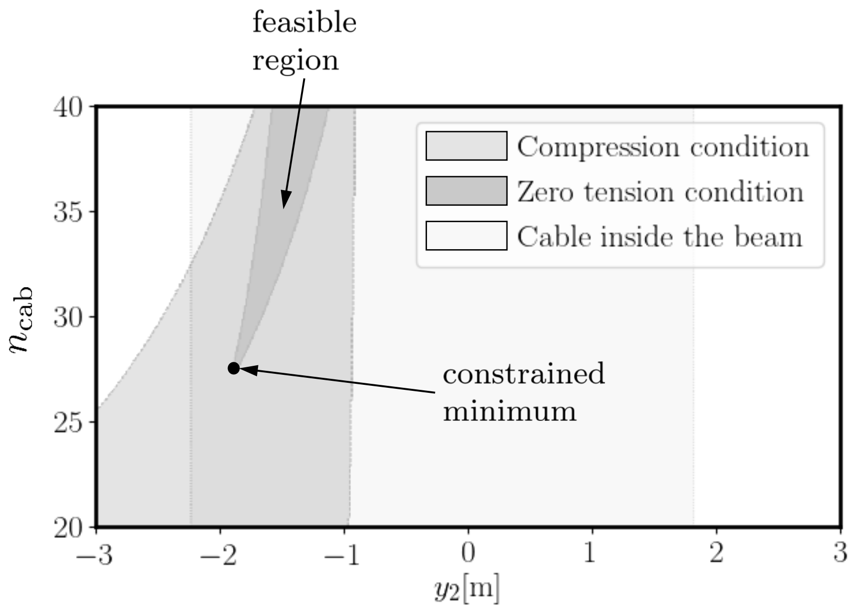


Fig. 6.6: The domain of the two dimensional problem.

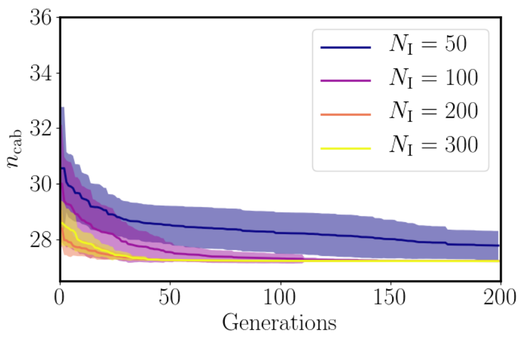
Number of individuals in a generation N_I

Four different values were tested. The process of the algorithm may be seen in the Fig. 6.7a, showing mean values of the best individuals in a generation from the ten runs performed. Standard deviation is also visible in the same, although slightly more transparent colour. In the Fig. 6.7b only the mean result of the ten runs along with the final standard deviation is shown.

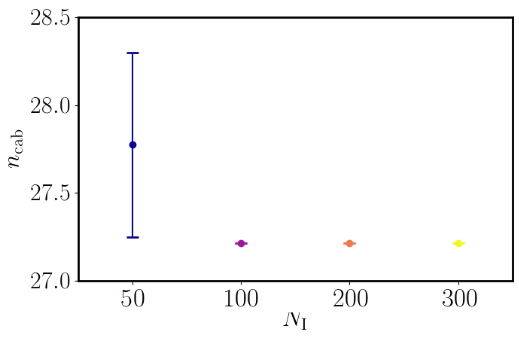
It is obvious, that too small population ($N_\mathrm{I} = 50$) have considerable effect on the algorithm. The convergence is in that case slower and the mean result is rather worse. The best convergence is obtained by $N_\mathrm{I} = 200$. However, the results of the other three values differ only slightly (mean results as well as standard deviation). For the reason of marginally faster convergence, for future application the number $N_\mathrm{I} = 200$ has been selected.

The number of individuals in the first generation kN_I

The aim of involving the multiplier k is to find a feasible individual in the first generation. From the graphs shown in Fig. 6.8a, it is visible that the algorithm found a feasible solution in the first generation in each of thirty runs (ten runs for each value). However, the value has no effect whatsoever of the result (Fig. 6.8b) as the difference between them is negligible. Having in mind, that the test case is only two-dimensional, a higher value of $k = 5$ has been selected for the benefit of future applications on more dimensional cases.

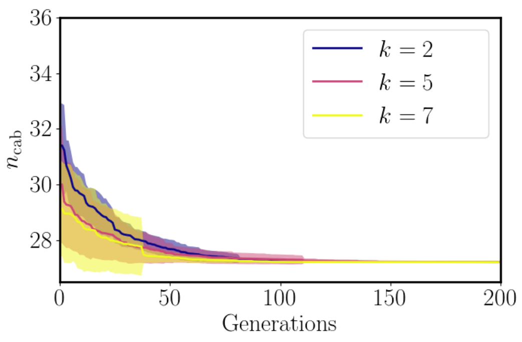


(a) Comparison of the convergence.

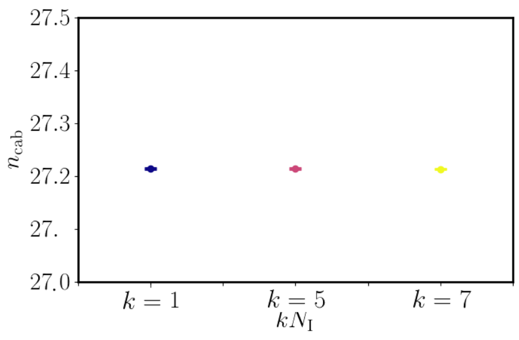


(b) Comparison of results.

Fig. 6.7: The performance of the algorithm for different N_{I}.



(a) Comparison of the convergence.

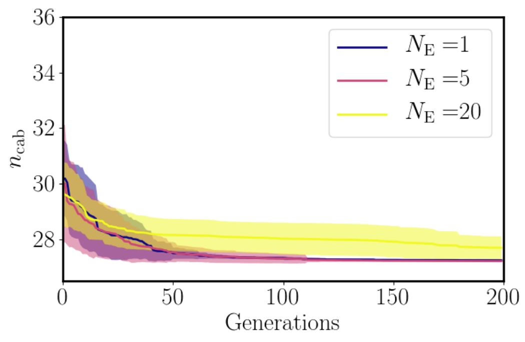


(b) Comparison of results.

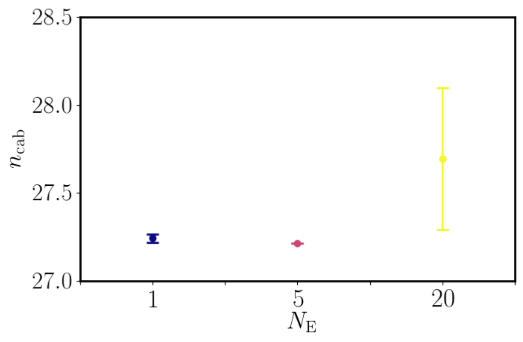
Fig. 6.8: The performance of the algorithm for different number of individuals in the first generation.

The number of elite individuals N_{E}

As already mentioned, the number of elite individuals suggested by De Jong in [6] is five. However, his research was performed on binary algorithm, thus the determination of the optimal value was performed as well. From the graphs (Fig. 6.9a) its observable that the large numbers of elite individuals perform poorly. The convergence in the case of $N_{\mathrm{E}} = 20$ is considerably worse and moreover, the algorithm is not capable of finding solutions of the quality of others (Fig. 6.9b). The results of the other two values are fairly similar, with the only difference being slightly more precise results in the case of $N_{\mathrm{E}} = 5$. For this reason, the value of five is used in future applications of the algorithm.



(a) Comparison of convergence.

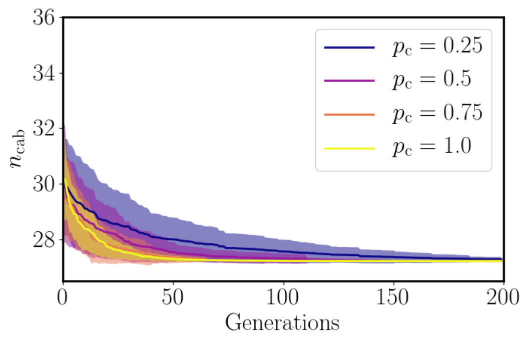


(b) Comparison of results.

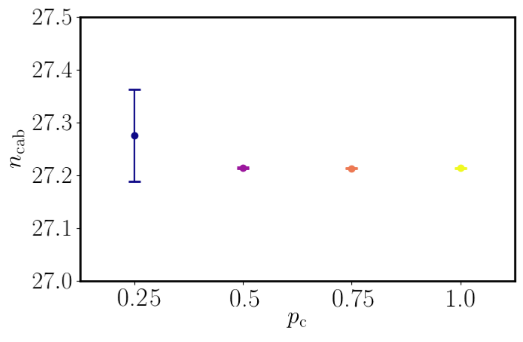
Fig. 6.9: The performance of the algorithm for different number of elite individuals N_E.

Crossover probability p_c

Based on the literature [12], usually the value of crossover probability ranges from 0.5 to 1. However, smaller value was tested as well and, as suggested by the literature, its result was poorer. Not only has the crossover probability significant effect on the convergence rate, but also on the quality of solutions, as depicted in the graphs (Fig. 6.10). When moving in the suggested range, only the convergence rate changes while quality of the solutions differs only slightly (Fig. 6.10b). The graph (Fig. 6.10a) shows, that the higher the crossover probability is, the faster the algorithm converges, therefore, the crossover probability is set to $p_c = 1$.



(a) Comparison of convergence.

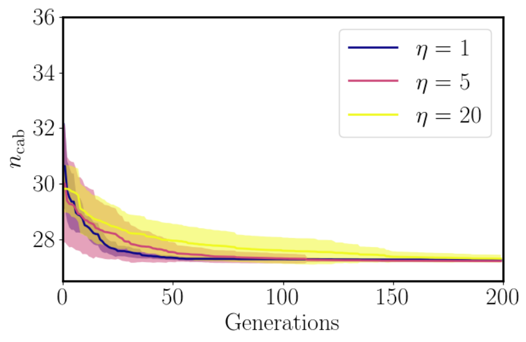


(b) Comparison of results.

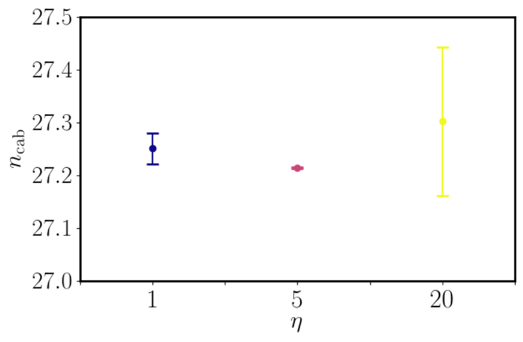
Fig. 6.10: The performance of the algorithm for different crossover probability p_c.

Crowding degree of crossover η

Three values of the crowding degree of crossover were tested, their performance may be seen in Fig. 6.11. Interestingly, their efficiency differs and favourable convergence rate is outweighed by poorer solutions and vice versa. Consequently, the middle value of $\eta = 5$ was selected. Despite the slower convergence, the quality of solutions is the best from the tested values.



(a) Comparison of convergence.



(b) Comparison of results.

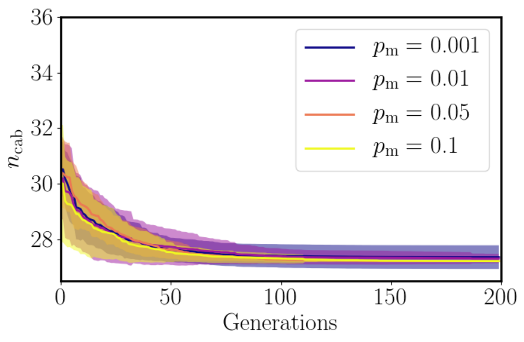
Fig. 6.11: The performance of the algorithm for different crowding degree of crossover η.

Mutation probability p_m

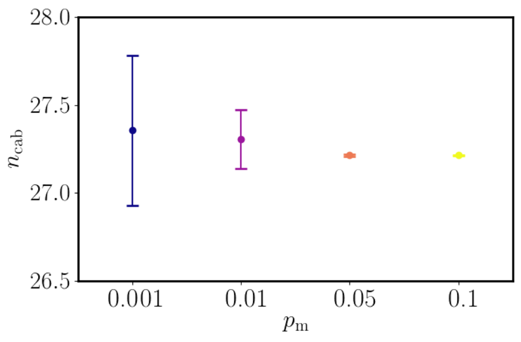
Four values of the mutation probability p_m were tested, the convergence and obtained solutions may be observed in the two graphs (Fig. 6.12). From the first graph it is apparent that the mutation probability does not play a significant role in convergence. However, standard deviation during the process differs considerably, suggesting that smaller values of p_m result in the inability of finding passable solution at each run. The results vary as well, but the same trend is visible and the highest mutation probability of tested values results in the best outcome. As such, in the future applications, the value of mutation probability $p_m = 0.1$.

The change of standard deviation of mutation through the computation

Six different combinations of the initial (σ_{ini}) and the final (σ_{fin}) standard deviation of mutation were tested. The premise is that in the beginning of computation, higher σ is desirable as it offers to mutate the member of the vector more radically. Thus, better exploration of the domain should be performed. A the end of the computation, the aim is exactly opposite: smaller σ is desirable to mutate the individuals only in the region already found.



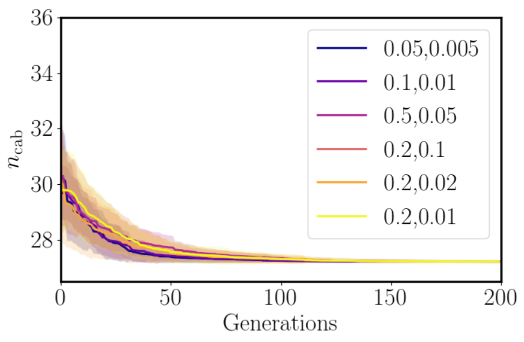
(a) Comparison of convergence.



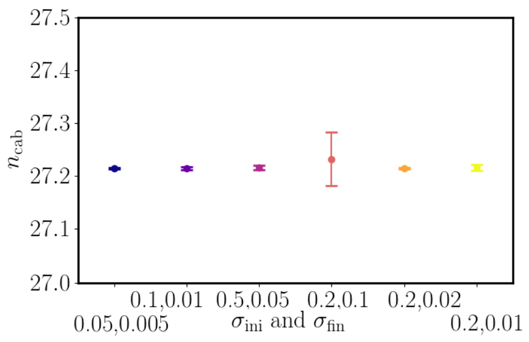
(b) Comparison of results.

Fig. 6.12: The performance of the algorithm for different mutation probability p_{m}.

From the graphs obtained by testing (Fig. 6.13), no significant influence of the values is apparent from the process. The only influence observable is that of $\sigma_{\text{fin}} = 0.1$ which results in a worse quality of the solution. However, it is possible that the sample is not sufficient to observe such a behaviour, with regard to both the number of generations and the number of runs performed. For future references, $\sigma_{\text{ini}} = 0.2$ and $\sigma_{\text{fin}} = 0.02$ is used and the decrease is linear.



(a) Comparison of convergence.



(b) Comparison of results.

Fig. 6.13: The performance of the algorithm for different σ_{ini} and σ_{fin}.

6.4 Optimization of tendons

Two examples were chosen for the application of the developed algorithm. The optimization of prestressed concrete tendon path of a simply supported beam and that of a continuous beam of two unequal spans.The formulation of the problem is defined as described in the section 6.1.2. The vector of design variables is formulated differently with regard to each problem.

Following characteristics were chosen. The cross-section is rectangular and constant, $b \times h = (0.5 \times 2)$ m, the length of the simply supported beam and the first span of the continuous beam $l_1 = 30$ m and the length of the second span of the continuous beam $l_2 = 20$ m. Concrete C30/37 with a characteristic compressive strength of $f_{\text{ck}} = 30$ MPa and specific weight of $\gamma_{\text{c}} = 25$ KN/m and a strand type of characteristic tensile strength $f_{\text{pk}} = 1860$ MPa was used.

The life span of the structure is 100 years. Two different loads are applied, each one in a different time. Creation time being day 0, self-weight is activated by prestressing on day 7 and an additional continuous load $g_1 = 30$ KN/m is applied on day 28. These forces result in different stress in the bottom and top fibers in each phase.

The genetic algorithm uses values determined in the previous section. Contrary to the test cases, number of generations $N_{\text{G}} = 500$ was used which along with number of individuals in a generation $N_{\text{I}} = 200$ and $k = 5$, adds up to the total number of evaluations $N_{\text{TOT}} = 100800$.

6.4.1 Simply supported beam

In the case of simply supported beam following vector of design variables was chosen with its respective domain:

$$\boldsymbol{x} = \{y_{1,3}, y_2, n_{\text{cab}}\} \tag{6.22}$$

$$-\frac{h}{2} \leq y_1 \leq \frac{h}{2} \tag{6.23}$$

$$-3 \leq y_2 \leq 3 \tag{6.24}$$

$$20 \leq n_{\text{cab}} \leq 40 \tag{6.25}$$

where $y_{1,3} = y_1 = y_3$ are the y coordinates of the first and the last control point of the Bézier curve. Because only the vertical load is applied, a symmetrical Bézier curve is used as adding another variable would complicate the process without offering better solutions. y_2 variable stands for y coordinate of the middle point. All three x coordinates of control points are fixed, $x_1 = 0$ m, $x_2 = \frac{l_1}{2}$ and $x_3 = l_1$. The last element of $f(\boldsymbol{x})$, n_{cab}, stands for the number of cabels neccessary for complying with given conditions(6.6, 6.7, 6.8). See the geometry with its variables in Fig. 6.14.

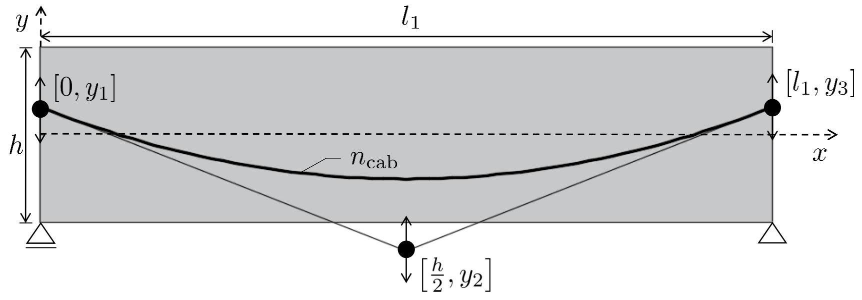


Fig. 6.14: Representation of the vector of design variables.

Again, ten runs were performed. The convergence of the algorithm can be observed on Fig. 6.15, where each colour represents one run and the labels stand for the resulting number of cabels n_{cab}. Because of a higher number of individuals in the first generation, in each case the algorithm found at least one feasible solution at the beginning. Furthermore, the algorithm converges quickly and from approximately the fiftieth generation the previously found solutions are exploited in order to find a more precise result.

Results obtained by genetic algorithm

From the solutions found by ten runs of the algorithm (Tab. 6.3, Fig. 6.16), it is noticeable that the difference between them is significant. Likely explanation is that the algorithm is not robust enough to find the global minima when more areas of local minima are present. The comparison of solutions is drawn in Fig. 6.17, where each solution is marked by the different colour as described by the labels. The same conclusion may be drawn from the same graph where values obtained by each solution are compared. Solutions which possibly found the same local minima are distinguished by horizontal lines.

Despite the differences, in each case the algorithm found a solution in which the maximum stress is exactly on the boundary of the condition (6.6) without exceeding it ($\sigma_{\mathrm{max}} = 0$ MPa), so no tensile stress appears through the beam's life span. Also, the maximum compression is well above the accepted percentage of the characteristic compressive strength of the chosen concrete ($\sigma_{\mathrm{min}} \geq 0.6 f_{\mathrm{ck}} = -18$ MPa). See Tab. 6.3 for the minimum and the maximum normal stress of all solutions.

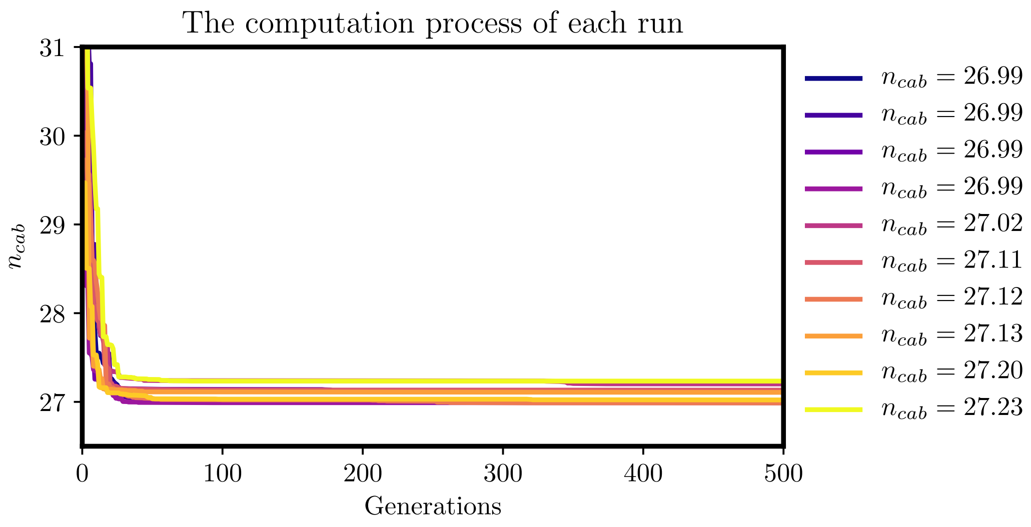


Fig. 6.15: The convergence of each computation by the algorithm.

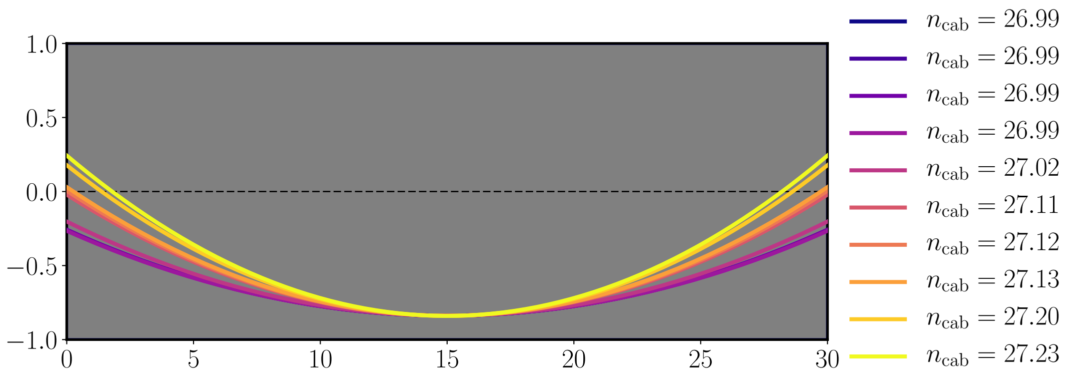


Fig. 6.16: Solutions found by the algorithm.

In the Fig 6.18 the normal stress the normal stress of the best solution is plotted. Bottom fibres are represented by continuous lines and top fibres by dashed lines and each colour represents different time. It is visible that at the time of creation on day 0, there is no normal stress present. Since prestressing on day 7 (represented by lighter blue), the self-weight is not sufficient to even the stress from prestressing. The bottom fibres are thus under compression and there is small compression up to no stress along the upper fibres. After applying a continuous load of magnitude $g_1 = 30$ KN/m on day 28, normal stress adequately shifts. We can observe that the top fibres are compressed, whereas there is no stress in the midspan of the bottom fibres and it is further decreasing towards the ends of the beam.

$y_{1,3}$ [m]	y_2 [m]	n_{cab}	$\sigma_{\max}$ [MPa]	$\sigma_{\min}$ [MPa]
-0.259	-1.417	26.99	0	-11.14
-0.259	-1.417	26.99	0	-11.14
-0.258	-1.418	26.99	0	-11.14
-0.266	-1.411	27.00	0	-11.14
-0.203	-1.474	27.02	0	-11.14
0.021	-1.657	27.11	0	-11.14
0.005	-1.683	27.12	0	-11.13
0.030	-1.708	27.13	0	-11.13
0.178	-1.857	27.20	0	-11.12
0.243	-1.923	27.23	0	-11.11

Tab. 6.3: Results of the optimization of the simply supported beam's tendon path sorted from the best solution to the worst.

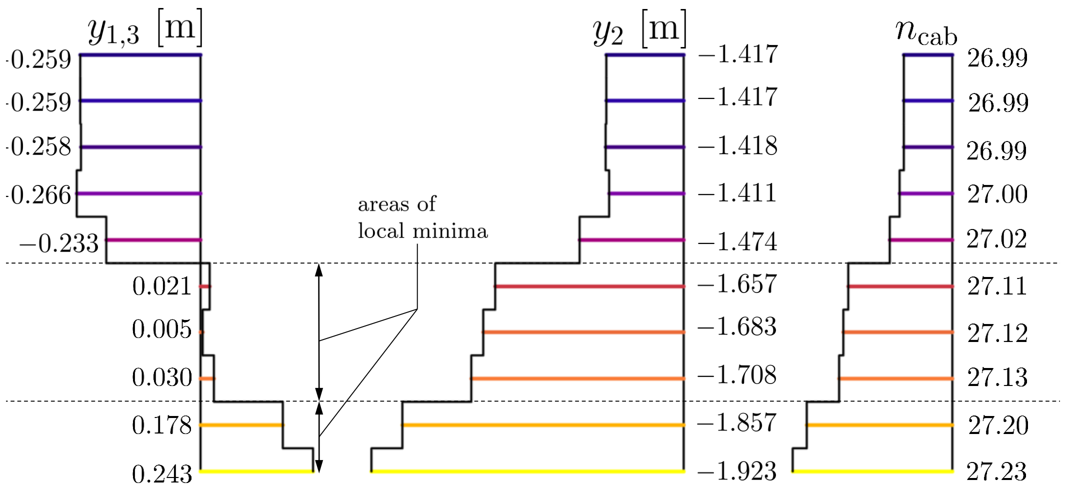


Fig. 6.17: Comparison of solutions found by the algorithm.

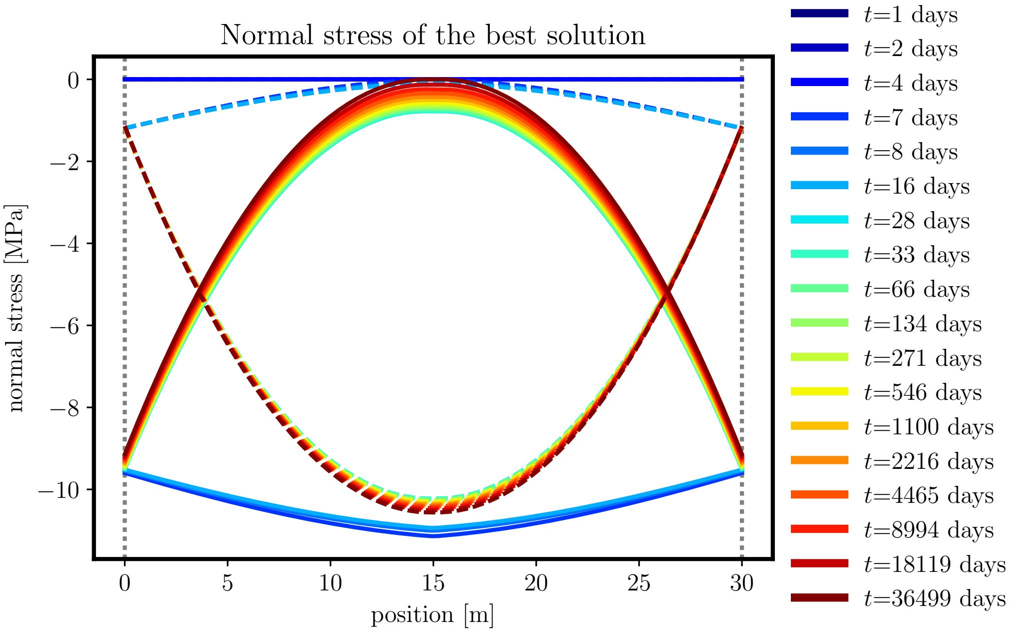


Fig. 6.18: Normal stress of the best solution.

6.4.2 Continuous beam of two unequal spans

For the optimization of a continuous beam of the two unequal spans, the problem is defined as described in the section 6.1.2. However, since two Bézier curves are present, one dimension to the vector of the design variables is added. The vector with its respective domain is formulated as follows:

$$\boldsymbol{x} = \{y_2, y_3, y_4, n_{\text{cab}}\} \tag{6.26}$$

$$-3 \leq y_2 \leq 0 \tag{6.27}$$

$$0 \leq y_3 \leq \frac{h}{2} \tag{6.28}$$

$$-3 \leq y_4 \leq 0 \tag{6.29}$$

$$10 \leq n_{\text{cab}} \leq 40 \tag{6.30}$$

Seeing that one variable was added, the range of the search domain was reduced to increase the chance of finding the optimal solution. See each variable in Fig. 6.19. The first point of the first Bézier curve and the last point of the second Bézier curve are fixed in both directions. The other thee points are flexible only in the direction of y-axis.

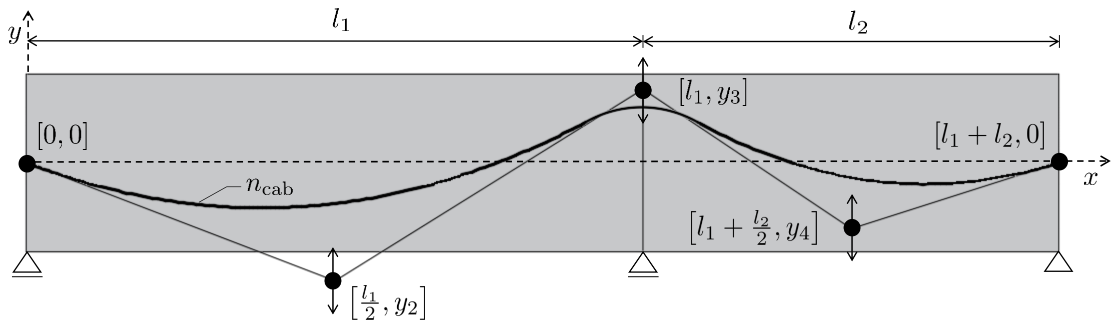


Fig. 6.19: Representation of the vector of design variables.

As a result of the restrained deformation by boundary conditions (the presence of the middle support) smaller bending moment in the middle of each span develops. Subsequently, the number of cables needed is expected to be considerably smaller in the case of the first span of continuous beam than in the case of the simple beam, even though both dimensions and loading are the same.

Resluts obtained by genetic algorithm

Ten runs of the algorithm were performed as well. The convergence of each computation by the algorithm is shown in Fig. 6.20. The similarities between the optimization of the simple beam and the continuous beam are evident: fast convergence in approximately the first fifty generations accompanied by the inability of finding the global minimum in the final stages of computation.

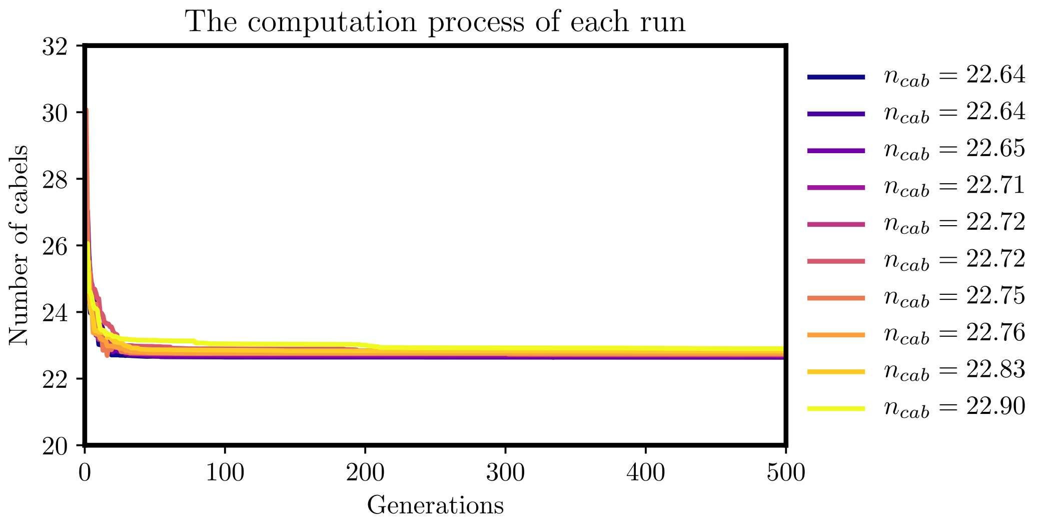


Fig. 6.20: The convergence of each run of the algorithm.

In each of the ten runs, the genetic algorithm found a solution, however, not the best solution was always achieved. In Fig. 6.21, all the tendon paths found by the algorithm are drawn. Unlike in the case of the simply supported beam, no local minima are apparent. The only visible trend is the compensation, as when one value is slightly greater, value of a different variable is smaller, see Fig. 6.22.

Nonetheless, even though the best solution and the worse solution differ by 0.26 cabel (see Tab. 6.4), each of the solutions found by the algorithm satisfy the constraints (6.6, 6.7, 6.8), and furthermore, each solution is positioned on the boundary of non-feasibility where $\sigma_{\max} = 0$ MPa. The best solution is on the edge of the beam, on the boundary of the third constrain. It can be also observed that the lower the Bézier curve lies, the better the solution is. In other words, the higher the Bézier curve lies, the more cables are needed to overcome tensile stress and subsequently, the higher compression occurs in opposite fibres.

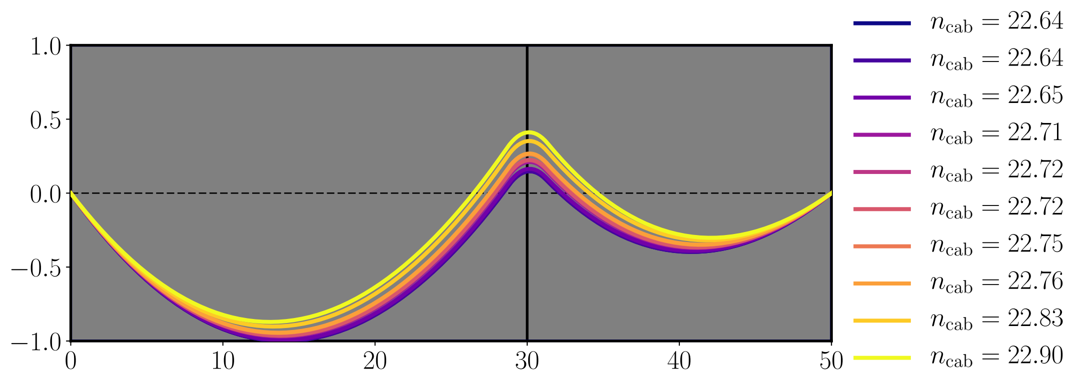


Fig. 6.21: Solutions found by the algorithm.

y_2 [m]	y_3 [m]	y_4 [m]	n_{cab}	σ_{max} [MPa]	σ_{min} [MPa]
-2.025	0.222	-0.839	22.64	0	-9.27
-2.025	0.223	-0.838	22.64	0	-9.27
-2.018	0.239	-0.831	22.65	0	-9.28
-1.993	0.290	-0.812	22.71	0	-9.31
-1.987	0.304	-0.807	22.72	0	-9.29
-1.989	0.300	-0.809	22.72	0	-9.29
-1.969	0.345	-0.789	22.75	0	-9.32
-1.967	0.350	-0.787	22.76	0	-9.32
-1.920	0.441	-0.757	22.83	0	-9.33
-1.882	0.504	-0.741	22.90	0	-9.30

Tab. 6.4: Results of the optimization of the continuous beam's tendon path sorted from the best solution to the worst.

Critical part after prestressing is in the top fibres of the first span and in the bottom fibres above the middle support (Fig. 6.23). In both cases there is no tension, however, stress is on the boundary of tension. After applying additional continuous load of 30 KN/m on day 28, top fibres above the middle support are almost under tension. Both fibres of the the second span are continuously compressed due to the smaller length of the span. However, it is clear that the algorithm has found solutions in accordance of given conditions and is capable of finding such solutions, that no tension occurs in each case.

In contrast to the optimization of the simply supported beam, the secondary prestressing effects due to the static indeterminacy of the beam occur [10].

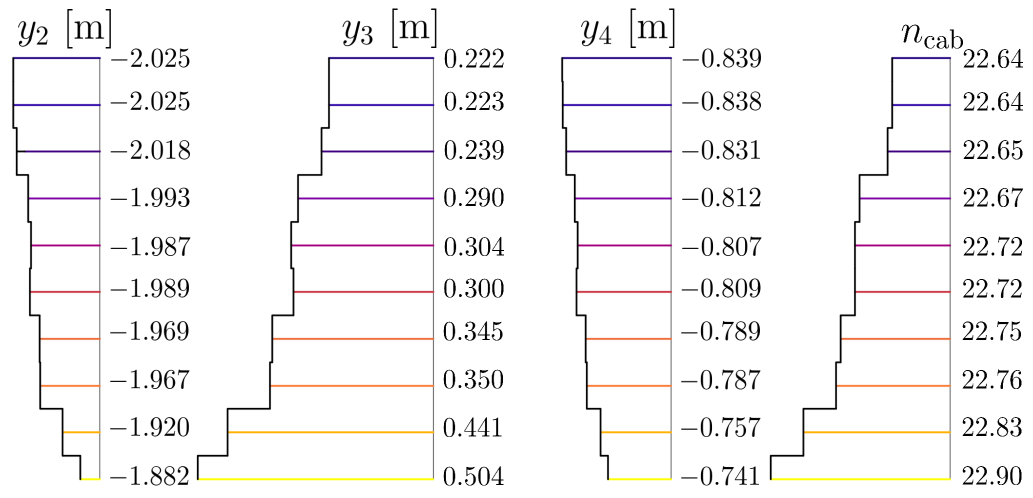


Fig. 6.22: Comparison of solutions found by the algorithm.

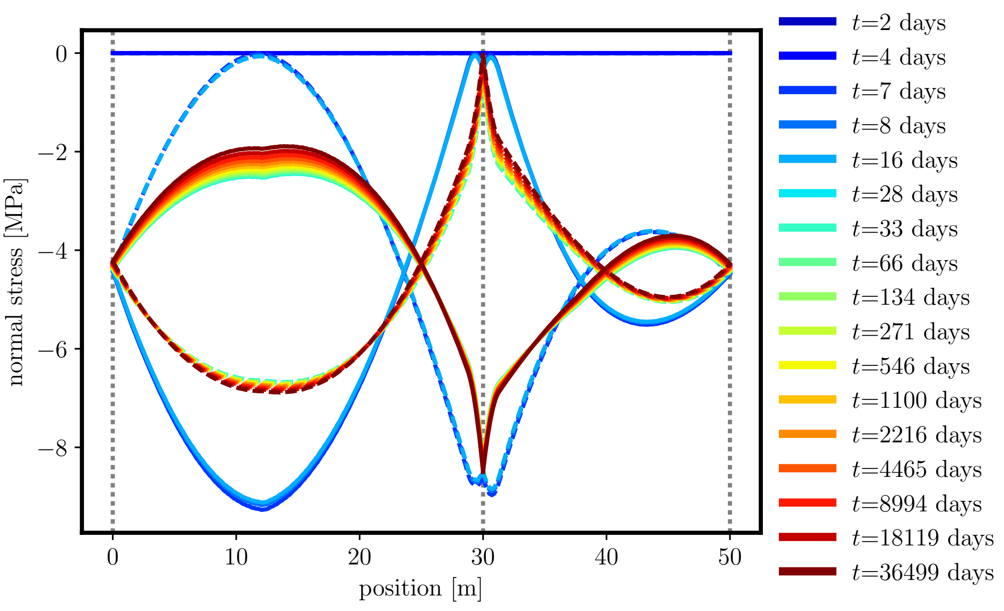


Fig. 6.23: Normal stress of the best solution.

7 Conclusion

The implementation of penalization function was tested on chosen analytical function and results were compared with a reference value. The penalization function has proven robust enough for the problem. Subsequently, the coefficients determining behaviour of the genetic algorithm were analysed. The results obtained by the analysis were then used for application of the algorithm to real problems.

Ten runs of optimization were performed on two problems: optimization of a prestressed concrete tendon path of the simple beam and that of the continuous beam of two unequal spans. The performance of the algorithm was similar in both cases when in the beginning the algorithm converged quickly only to search in the already found regions for the rest of the computation.

The life span of optimized beams was set to 100 years with two loads applied in different times: prestressing on day 7 and additional continuous load on day 28. By applying the additional load on day 28, the normal stress in the element shifts and different stress occurs in bottom and top fibres, making the constraints more restrictive.

In the case of the simply supported beam optimization, the algorithm found varying solutions. Graphical representation of obtained solutions suggests, that at least two important local minima are present. Nevertheless, the algorithm has always found solutions complying with given constraints. Furthermore, the condition of zero tension has proven as the most restrictive. All the solutions found by the algorithm were on the boundary of the constraint, where maximal stress equals zero.

The algorithm performed similarly in the case of continuous beam. A solution complying with given constraints was found each time. As in the previous case, each solution was positioned on the boundary of non-feasibility of the same constraint. However, the solutions found by the algorithm differed between the ten runs. Possibly several local minima are present in the domain, whose values are close and the algorithm might not be robust enough to overcome them.

The algorithm was developed in the Python programming language with the aid of its built-in libraries.

Bibliography

[1] SEDERBERG, Thomas W.: *Computer aided geometric design.* Birgham, 2016. Course notes. Birgham Young University.

[2] FORTIN, Félix-Antoine, François-Michel DE RAINVILLE, Marc-André GARDNER, Marc PARIZEAU, Christian GAGNÉ: *DEAP: Evolutionary Algorithms Made Easy.* Journal of Machine Learning Research. 2012, 2171-2175.

[3] GOLDBERG, David E.: *Genetic algorithms in search, optimization, and machine learning.* Reading, Mass.: Addison-Wesley Pub. Co., 1989. ISBN 978-0201157673.

[4] HAUPT, Randy L.; S. E. HAUPT: *Practical genetic algorithms.* 2nd ed. Hoboken, N.J.: John Wiley, 2004. ISBN 978-0471455653.

[5] HOLLAND, John H.: *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* Ann Arbor: University of Michigan Press, 1975. ISBN 9780472084609.

[6] DE JONG, Kenneth Alan: *Analysis of the behavior of a class of genetic adaptive systems.* Michigan, 1995. Dissertation thesis. University of Michigan.

[7] MICHALEWICZ, Zbigniew: *Genetic Algorithms + Data Structures = Evolution Programs.* 2nd. Berlin: Springer, 1992. ISBN 978-3-662-03315-9.

[8] MITCHELL, Melanie: *An introduction to genetic algorithms.* Cambridge, Mass.: MIT Press, 1996. ISBN 02-621-3316-4.

[9] MONTEMURRO, Marco, Angela VINCENTI and Paolo VANNUCCI: The Automatic Dynamic Penalisation method (ADP) for handling constraints with genetic algorithms. *Computer Methods in Applied Mechanics and Engineering.* 2013(256), 70 - 87. DOI: https://doi.org/10.1016/j.cma.2012.12.009. ISSN 0045-7825.

[10] NAVRÁTIL, Jaroslav: Prestressed concrete structures. Brno: Akademické nakladatelství CERM, 2006. ISBN 80-720-4462-1.

[11] VAN ROSSUM, G.: *Python tutorial, Technical Report CS-R9526*, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, May 1995.

[12] SIMON, Dan: *Evolutionary optimization algorithms: biologically-Inspired and population-based approaches to computer intelligence.* Hoboken, New Jersey: John Wiley, 2013. ISBN 978-0-470-93741-9.

[13] SINGIRESU S. RAO: *Engineering optimization theory and practice.* 4th ed. Hoboken, N.J: John Wiley, 2009. ISBN 9781615831791.

[14] BAŽANT, Z. P. a Robert L'HERMITE. *Mathematical modeling of creep and shrinkage of concrete.* New York: Wiley, 1988. ISBN 0471920576.