18th International Conference
ENGINEERING MECHANICS 2012
Svratka, Czech Republic, May 14 – 17, 2012

pp. 1049–1059
Paper #230

BRANCH AND BOUND METHOD FOR GLOBAL OPTIMA OF SIZE OPTIMIZATION BENCHMARKS

A. Pospíšilová [*], **M. Lepš** [**]

Abstract: *This contribution focuses on searching for global optima of size optimization benchmarks utilizing a method based on branch and bound principles. The goal is to show the process of finding these global optima on several examples. To minimize computational demands a suitable parallelization is used. Optima which can be found in available literature and optima obtained in this work are compared.*

Keywords: ***benchmarks, discrete sizing optimization, branch and bound method, global optima, parallel programming***

1. Introduction

A numerical optimization is nowadays a very popular tool for obtaining a different view on structures and materials. Shape of the structure, cross-sections, amount of reinforcement, thicknesses of sheets, design of concrete mixture and many other properties can be optimized. Recently, many heuristic algorithms are developed, tested on benchmarks and their efficiency is compared. To compare distinct optimization methods, it is appropriate to know the global optima of these benchmarks. The closer to the global optimum the gained value is, the better the method is. In the past, it was not possible to obtain these optima because of large computational demands. A computational power is growing every year therefore now seems to be the right time to deal with this issue.

This paper is trying to outline a process of searching for global optima of sizing discrete optimization benchmarks. Various optimization methods can be used for obtaining optima such as gradient methods (Shewchuk (1994)), heuristics methods (Dréo et al (2005)), or evolutionary algorithms (Eiben and Smith (2003)). These methods do not guarantee that the gained optimum is the global one because only a portion of the space is explored. Nevertheless, the advantage of these methods is that the optimum is found in a real time and the ability to obtain or at least approach a vicinity of a global optimum is considered as a sign of quality. In our work, we used a method based on branch and bound principles to obtain global optima and appropriate cross-sections. A good estimate of a lower and upper bounds reduces the searched space but still ensures that the global optima can be found. The algorithm presented in our paper is universal, i.e. it is applicable to other truss structures or a similar type of problems. We hope that the knowledge of global optima of studied benchmarks will improve the development of optimization methods.

2. Sizing optimization

Sizing optimization (Bendsoe and Sigmund (2003)) is one type of structural optimization that deals with truss-like structures. These structures are defined by topology, material, loading, supports, and a set of sections or alternatively minimum and maximum cross-sectional areas of the individual rods. The objective function is the weight of a structure and constraints are maximal stresses and maximal displacements, respectively. The goal is to find cross-sections for the given structure that satisfy prescribed constraints and have a minimal weight. The selection of cross-sections from the given set of sections defines a discrete optimization problem, whereas variables chosen from given limits leads to a continuous case. The

[*]Ing. Adéla Pospíšilová: Faculty of Civil Engineering, CTU in Prague, Thákurova 7; 166 29, Prague; CZ, e-mail: adela.pospisilova@fsv.cvut.cz
[**]Ing. Matěj Lepš, Ph.D.: Faculty of Civil Engineering, CTU in Prague, Thákurova 7; 166 29, Prague; CZ, e-mail: leps@cml.fsv.cvut.cz

continuous optimization problem can be efficiently solved by mathematical programming methods like gradient-based methods as will be shown later in the text. When using discrete variables, no such option is available. Thus our attention is aimed at the discrete case.

3. Discrete optimization problem

The goal is to find such combination of cross-sections from the given list of profiles that leads to minimal weight still fulfilling given constraints. Here, two methods that are able to find global optima for this discrete optimization problem are presented.

3.1. Enumeration

An *Enumeration* is the simplest method for obtaining a global optimum of the discrete optimization problem. It is necessary to compute values of an objective function and constraints for every combination of cross-sections from a given set. Therefore, the enumeration has very large computational demands. If there are n sections and k variables (i.e. rods or groups of rods) than n^k possible solutions exist, i.e. the problem grows exponentially with a growing number of variables. The application of the enumeration is therefore possible only for small structures or for analysis of the vicinity of some local optima.

3.2. Method based on branch and bound principles

A *branch and bound method* is another method for obtaining global optima. A. M. Land and A. G. Doig (see Land and Doig (1960)) invented this method for linear problems. It was modified for discrete problems and for mixed-discrete variables (see e.g. Arora (2002)) many times.

A branch and bound method is based on a division of a main problem to several subproblems, so-called branches. To estimate, which branches are to be evaluated, an existence of the lower and upper bounds needs to be assumed, i.e. the lower and upper bounds are used to restrict the searched space. The lower bound can be obtained by any continuous optimization method, because the global optimum with discrete design variables will never provide a lower value of the objective function than the global optimum with continuous design variables. The upper bound can be obtained by any heuristic method, because a local optimum always has a greater or equal value of the objective function than the global optimum. Since the constraints for the sizing optimization problem are more computationally demanding than the value of the objective function, they are calculated only for solutions that lie between lower and upper bounds. If we obtain a subproblem with a value of the objective function outside the given bounds, the rest of a branch is not calculated because global optimum cannot be located there. The more accurate estimates of the lower and upper bounds are, the narrower the searched space can be. Especially, the upper bound can be decreased during the calculation based on the already obtained objective function's values. Hence, the searched subspace will be reduced and a problem will be solved with less computational demands.

4. Continuous optimization problem

A continuous optimization problem is more complex than the discrete one because an infinite number of potential solutions exists in the space with real numbers. Therefore, it cannot be guaranteed that the found optimum is the global one. Nevertheless, it is possible to use powerful continuous optimization algorithms such as mathematical programming methods which are well established. Obtaining a potential global optimum with continuous variables is therefore less demanding than the solution of the optimization problem with discrete variables. The main disadvantage of this methodology is the uncertainty of a solution quality. It can be overcome by the following alternatives:

- The branch and bound method expects that the lower bound has the same (or higher) value of the objective function as the global optimum with continuous variables. Since the global optimum of the continuous problem cannot be generally known, the true lower bound cannot be ensured. As a solution, the lower bound is set to its lowest potential minimum i.e. without using any continuous optimization method. This process provides a real global optimum with discrete variables. In most

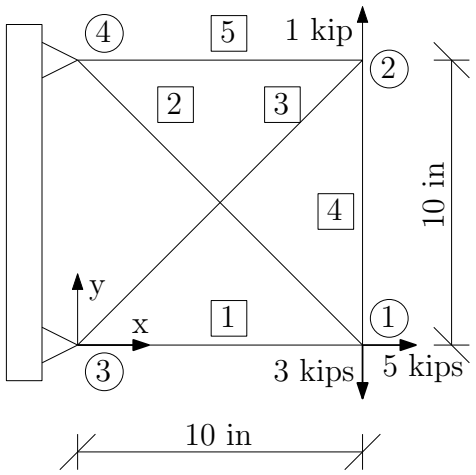

Fig. 1: The 5-bar truss

cases, however, the searched space will be extremely huge for computation of all possible solutions in a real time.

- Another approaches do not fully guarantee the acquisition of the global optimum. Nevertheless, the probability of obtaining the global optimum is acceptable. These approaches are based on an estimation of the global optimum with continuous variables and the value of its objective function. We use nonlinear programming that is implemented in MATLAB environment (e.g. `fmincon()` function). This routine is executed several times from random initial points. If the obtained optima do not differ from each other and the results are comparable to optima published in available literature, the estimate is considered as credible. If the gained optima differ from each other, then it is not possible to use them as the lower bound. The first approach (without using the continuous optimization method) is then used or the lower bound is estimated to be e.g. 20% lower than the best-gained value.

All continuous optima for problems mentioned below in this text were consistent with published optima. For the sake of certainty, the nonlinear programming method was launched with different starting vectors hundred times and the best solution was considered as the the lower bound.

5. Sizing optimization benchmarks

5.1. 5-bar truss

A representative example of a structure small enough for computational demands and bigger enough for branching purposes was necessary for developing the branch and bound algorithm. The topology of the structure was taken from reference Lee and Hajela (2001), constraints and a set of given cross-sections were chosen by authors.

A structure in Fig. 1 has four nodes and five rods and is made from aluminium. The density of the material is $0.1\ lb/in^3$ and Young's modulus is equal to $10^4\ ksi$. The allowable stress is limited to $\pm 60\ ksi$ in each rod and the displacements are limited to $\pm 0.06\ in$ along the horizontal and vertical directions. Continuous variables can assume values between lower $0.01\ in^2$ and upper bounds $0.1\ in^2$, respectively. Note that imperial units are used in the whole text because the gained solutions will be compared with published optima in available literature where imperial units are usually used.

A function of nonlinear programming `fmincon()` (The MathWorks (2011c)) offers four variants of optimization algorithms. In this paper, a Sequential quadratic programming (SQP) is a suitable algorithm for continuous sizing optimization problem. It is chosen by a command `options = optimset('Algorithm','active-set')`. Main idea is to convert a more complicated problem to a problem that is easier to solve. Here, the constrained nonlinear problem is solved using a sequence of parameterized unconstrained optimizations which, in the limit (of the sequence) converge to the constrained problem (The MathWorks (2011a)).

A starting point, i.e. a design variable vector composed of cross-sectional areas, an objective function, constraints and lower and upper bounds of variables are necessary as an input at the beginning of the algorithm. The objective function is a weight of the structure

$$m = f(A_i) = \rho \cdot \sum_{i=1}^{N} A_i \cdot L_i, \tag{1}$$

where $N = 5$ is a number of rods, A_i is a cross-sectional area and L_i is the length of a rod i and ρ is density of the material. Constraints are defined as inequalities such as

$$\max|\sigma_i| - 60 \leq 0, \tag{2}$$

$$\max|w_j| - 0.06 \leq 0, \tag{3}$$

where $\max|\sigma_i|$ is a maximal absolute value of stresses, j is an ordinal number of independent displacements and $\max|w_j|$ is a maximal absolute value of displacements. These values can be obtained by several methods. In this paper, the finite element method was used as is described e.g. in Pospíšilová (2010).

You can see the results for the continuous optimization problem of the 5-bar truss in Tab. 1. The objective function value of the optima gained with SQP algorithm is later used as the lower bound for the branch and bound method.

An identical topology of the 5-bar truss is used for the discrete optimization version. Material properties and constraints are also identical. The cross-sectional areas are chosen from the set $\{0.01, 0.02, \ldots, 0.1\}$ in^2. Since the structure has five rods ($k = 5$) and 10 cross-sectional areas ($n = 10$), the number of all possible solutions is $n^k = 10^5$. Therefore, the structure is small enough and the discrete global optimum can be obtained with the enumeration. The results obtained by the enumeration can be seen in Tab. 1.

The lower bound for the branch and bound method is set to the optimum value of the objective function obtained with continuous variables. The upper bound is set to the estimated weight 0.23 lb that is 25% greater than the global optimum value of the objective function gained by the enumeration. A space is searched systematically between these two bounds until the global optimum is found.

The steps of the algorithm can be described as follows:

1. First of all we have to decide which values will be used as initial. It is appropriate to begin with the lowest profiles and increase them because of minimization of the objective function. From a programming point of view, it is easier to use integer variables that are the ordinal numbers of the given set of cross-sectional areas - set M. For example, the initial design variable vector is `1 1 1 1 1`, which means that the first area (0.01 in^2) from the given set is attached to each rod. For numbering of rods see again Fig. 1.

2. The value of the objective function (a weight of a structure m) is then calculated and compared with the lower m_{min} and upper m_{max} bounds. If the weight of the structure is less than m_{min}, the algorithm will go to Step 3. If the structure weight is between m_{min} and m_{max} the algorithm will go to Step 4. If the weight of the structure is greater than m_{max}, the algorithm will go to Step 5.

3. The value of the objective function is less than m_{min}. It is necessary to find design variables' combination with greater weight than m_{min}. The last variable is raised to its maximum for faster progress of the algorithm as `1 1 1 1 10` and the value of the objective function is calculated and compared with m_{min}.

   (a) If the value of the objective function is still less than m_{min}, the algorithm searches for the design variables' combination with greater weight than the lower bound. This can be done as follows. The next-to-last variable is repeatedly raised by one, e.g. to (`1 1 1 2 10`). If the next-to-last variable value reaches its maximum it is decreased to its minimum and the third from the end variable value is raised by one. The algorithm will goes to Step 2 at the moment when all variables are set such that $m > m_{min}$.

Tab. 1: 5-bar truss optima

Variable	Units	Discrete optimization		Continuous optimization
		Enumeration	Branch and bound method	`fmincon()`
A_1	in^2	0.05	0.05	0.0500
A_2	in^2	0.01	0.01	0.01
A_3	in^2	0.06	0.06	0.0471
A_4	in^2	0.02	0.02	0.0167
A_5	in^2	0.01	0.01	0.01
m	lb	**0.179**	**0.179**	0.157
$\max \lvert w_j \rvert$	in	0.059	0.059	0.06
$\max \lvert \sigma_i \rvert$	ksi	59.371	59.371	60.006
w_{lim}	in	0.06	0.06	0.06
σ_{lim}	ksi	60	60	60

(b) If the value of the objective function is greater than the lower bound, the last variable value is decreased to its minimum (`1 1 1 1 1`) and is increased one by one (`1 1 1 1 2`, `1 1 1 1 3`, ..., etc.) until the weight is greater than m_{min}. If $m_{min} > m$ the algorithm goes to Step 2.

4. The value of the objective function is greater than m_{min} and less than m_{max}. The global optimum is located somewhere in this subspace. Therefore, the constraints are evaluated, i.e the stresses and displacements are calculated.

   (a) If the constraints are fulfilled, i.e. $\max \lvert \sigma_i \rvert \leq 60\ ksi$ and $\max \lvert w_j \rvert \leq 0.06\ in$, the upper bound is updated to the actual objective function value $m_{max} = m$. Thus the upper bound is pushed down towards the global optimum and the searched space is reduced. The last variable value is increased by one afterwards. If this variable value exceeds its maximal possible value of the cross-sectional area from a given set, e.g. `1 1 5 11 1`, its value is set to the minimal possible value and the next-to-this variable value is increased by one, i.e. `1 1 6 1 1`. The algorithm goes to Step 2.

   (b) If the constraints are not fulfilled, the value of the last variable is increased by one.

5. The value of the objective function is greater than m_{max}. The value of the last variable is decreased to its minimum, the next-to-last variable value is increased by one and the objective function value is calculated. If the variable exceeds its maximal possible value from the given set the algorithm acts as in Step 4a.

   (a) If the value of the objective function m is lower than m_{max}, the algorithm goes to Step 2.

   (b) If the objective function value m is greater than m_{max}, a value of the third variable increases by one and the value of the forth variable is set to its minimum. The algorithm continues in this way until the objective function value is less than m_{max}. If there is no such combination of cross-sectional areas, the task is terminated.

6. If all variable values are set to their maxima, the algorithm ends.

Fig. 2 shows a distribution of 5-bar truss potential solutions. A seagreen part shows a number of potential solutions below the lower bound where only the objective function values are calculated, i.e.

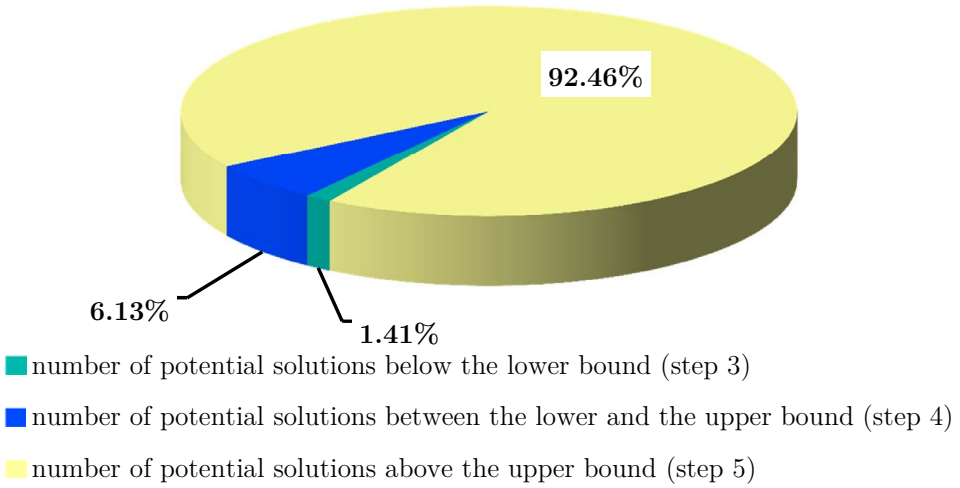

Fig. 2: A pie chart of a distribution of the 5-bar truss problem solutions solved by the branch and bound method

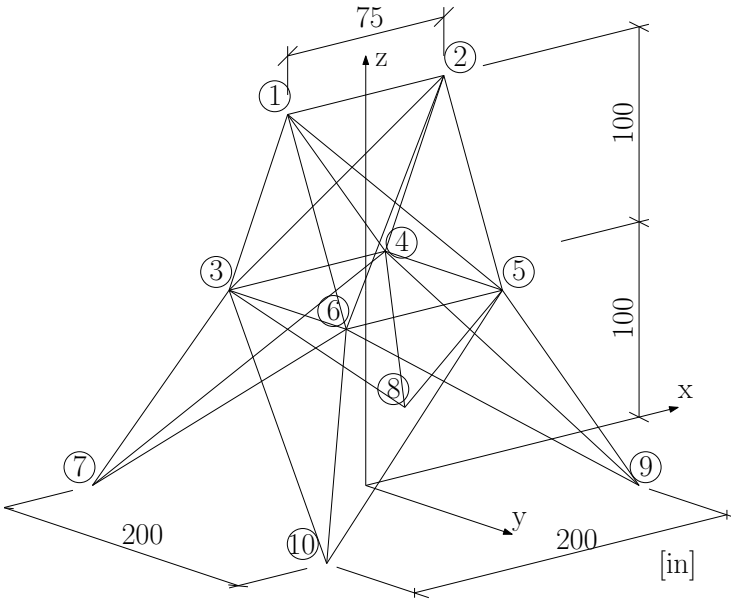

Fig. 3: The 25-bar truss

Step 3 of the algorithm. A blue part shows a number of potential solutions between the lower and the upper bound where values of the objective function as well as constraints are calculated. See Step 4 of the algorithm for more details. The global optimum is included in this subspace. A yellow part represents a number of potential solutions above the upper bound where only the objective function values are calculated. See Step 5 of the algorithm for more details. Tab. 1 presents results for the continuous optimization problem along with the results for the discrete problem solved by the enumeration and the branch and bound method. Since the enumeration calculates values of the objective function as well as constraints for all potential solutions it is not possible to omit the global optimum. Results obtained by both presented methods are identical and this comparison serves as verification of the branch and bound method.

5.2. 25-bar truss

A topology of this test problem has been firstly used in reference Fox and Schmit (1966). The structure has ten nodes and four supports (see Fig. 3); therefore there are 18 free displacements. The structure is symmetric thus some rods were linked to groups, listed in Tab. 2. The material is aluminium with density equal to $0.1\ lb/in^3$ and Young modulus equal to $10^4\ ksi$. The loading is defined in Tab. 3. Every

Tab. 2: Member grouping for the 25-bar truss

Group of bars	Conectivities
A_1	1-2
A_2	1-4, 2-3, 1-5, 2-6
A_3	2-5, 2-4, 1-3, 1-6
A_4	3-6, 4-5
A_5	2-4, 5-6
A_6	3-10, 6-7, 4-9, 5-8
A_7	3-8, 4-7, 6-9, 5-10
A_8	3-7, 4-8, 5-9, 6-10

Tab. 3: Loadings for the 25-bar truss (kips)

Node	F_x	F_y	F_z
1	1.0	-10.0	-10.0
2	0	-10.0	-10.0
3	0.5	0	0
6	0.6	0	0

cross-sectional area is chosen from the given group: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.8, 3.0, 3.2 and 3.4 in^2, see reference Wu and Chow (1995). Continuous variables can assume values between a lower 0.1 in^2 and an upper bound 3.4 in^2, respectively. The allowable stress is set to ± 40 ksi in all rods and maximal allowable displacement is ± 0.35 in at all nodes along the x, y and z directions.

The same methodology was used for the 25-bar continuous optimization case as for the 5-bar truss problem. The results are shown in Tab. 4 and they are compared with the results published in available literature. The discrete case cannot be enumerated in a reasonable time because the number of potential solutions is $n^k = 30^8 = 6.561 \cdot 10^{11}$, where k is a number of rod groups. The discrete global optimum was gained with the branch and bound method, where the lower bound was set to the gained optimum from continuous optimization and the upper bound was set as the worst available solution from literature (see Rajeev and Krishnamoorthy (1992)). The algorithm is the same as in the 5-bar truss problem.

6. Parallelization

The 25-bar truss is relatively computationally demanding. Since the evaluation of solutions is independent to each other (except updating the upper bound m_{max} described in Step 4a of the algorithm), it is possible to run the method in a parallel way. Nowadays, modern computers are equipped with several core processors and thus it is appropriate to use this computational effort. MATLAB environment offers several tools of parallelization which are presented in this section.

The simplest parallelization method is to permit usage of all cores [1] by the `maxNumCompThreads()` command, see Luszczek (2009) for more information. There is no need to change a serial code; MAT-

[1] It is possible to permit only a subset of all available cores such as two cores out of four.

Tab. 4: Comparison of results for the 25-bar truss continuous case

Variable	Unit	Perez & Behdinan	this paper
		2007	2011
A_1	in^2	0.1	0.1
A_2	in^2	0.457	0.421
A_3	in^2	3.4	3.4
A_4	in^2	0.1	0.1
A_5	in^2	1.937	1.917
A_6	in^2	0.965	0.966
A_7	in^2	0.442	0.471
A_8	in^2	3.4	3.4
m	lb	483.84	**483.82**
$\max \lvert\sigma_i\rvert$	ksi	6.15	6.13
$\max \lvert w_j\rvert$	in	0.35	0.35
σ_{lim}	ksi	40	40
w_{lim}	in	0.35	0.35

LAB does all parallelization by itself. This variant of parallelization is not the best one because the user cannot mark appropriate parts for parallelization and also shared memory cannot be accessed.

Another possibility is to use a `parfor` loop instead of `for`. Every iteration of a `parfor` loop is independently executed on individual cores. For proper functioning of the `parfor` loop, it is necessary to prepare an appropriate number of processes/threads, so-called *labs*, by command `matlabpool open N`, where `N` is the number of opened labs. This reserves a collection of MATLAB worker sessions to run the loop iterations. The number of labs can be equal to the number of cores or less. Every lab has own part of the `parfor` loop. However, the user cannot specify any partition of data. It is also important to know that there is also no possibility to work with shared memory. Therefore it is not possible to update the upper bound value efficiently and therefore, this way of parallelization is not useful for the branch and bound method.

The last possibility mentioned here is the `spmd` method which means Single Programm Multiple Data, see e.g. The MathWorks (2011b) for more details. The `spmd` statement separates the block of a code to be run simultaneously on multiple labs. As well as in the `parfor` loop method, the command `matlabpool open N` open a required number of labs. Sending data to another lab is possible by the `labSend(data, X)` command, where `X` is the index of receiving lab where the data are sent. Then, it is necessary to receive the data by the `labReceive(Y)` command, where `Y` is the index of a lab from which the data will come. It is appropriate to split the data only at one, so-called *master*, lab and receive data with the others, so-called *slaves*. The master can process its own data as well.

The main problem here is to estimate a proper amount of data for every lab. If the data are sent too often the communication between the master and the slaves will take a plenty of time. In the 25-bar truss task, permutations with repetition are generated in advance for several groups of rods (i.e. four groups) and the remaining groups of rods (other four groups) are generated in the branch and bound method independently on each labs. In advance generated combinations are divided in the `for` loops to individual labs and then the algorithm continues as in the algorithm written for the 5-bar truss task. The maximal values of m_{max} are collected at the end of every iteration. The smallest one is chosen as a new

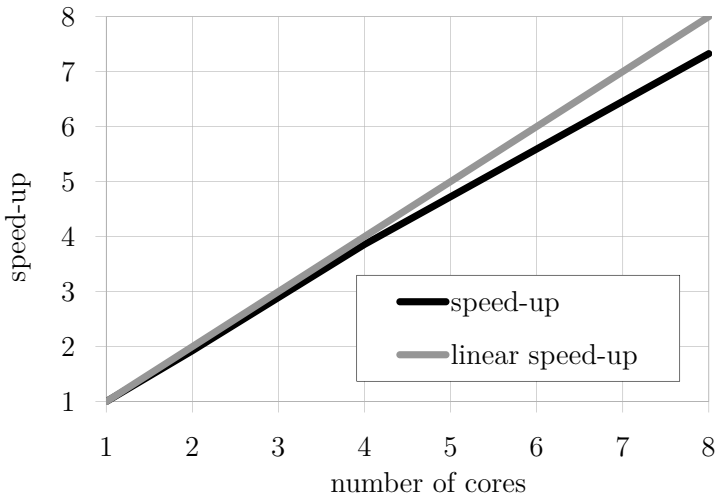

Fig. 4: A speed-up of the 25-bar truss problem solved by the parallel branch and bound method

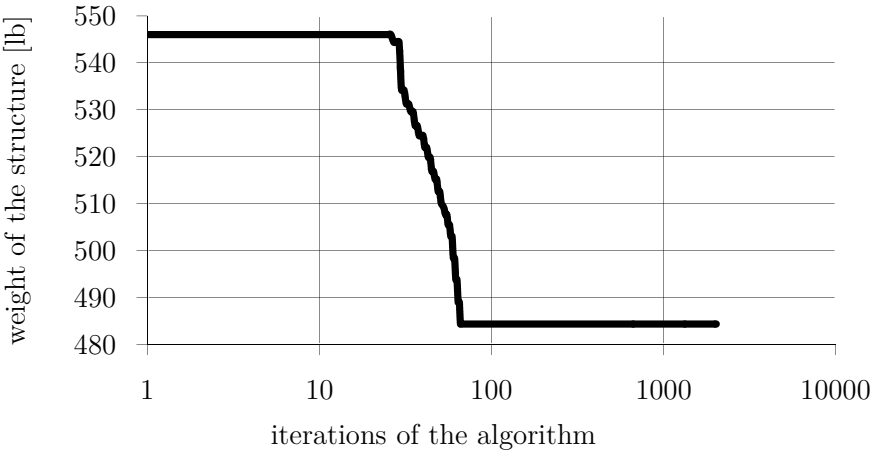

Fig. 5: A graph of the decreasing upper bound for the 25-bar truss problem

m_{max} and resent to every lab as an initial value of m_{max} for another iteration. If all data are used, the smallest value of m_{max} is taken as the global optimum.

For the parallel version of the algorithm, it is important how well the task is scaled, whether the parallelization is useful or not. Ideally, we would like to achieve linear scaling i.e. speed-up of n on n cores. However, it is very hard to obtain linear scaling, e.g. because of time spent on communications. Fig. 4 shows a graph where speed-up of the parallel algorithm is compared on 1 to 8 labs[2]. HP Xeon Z600 Workstation with two 4-cores processors Intel Xeon E5520, frequency 2.27GHz was used for computations within Matlab R2009a 64-bit in Debian GNU/Linux.

7. Conclusions

Fig. 5 shows a graph with the decreasing upper bound m_{max} for the 25-bar truss problem. The value of m_{max} determines the best solution in a progress of the algorithm. It can be interpreted as a convergence of the objective function to the global optimum. In advance generated combination were sent to eight labs by fifty. The number of iterations were $30^4/(8 \cdot 50) = 2025$. The global optimum was gained in the 66^{th} iteration. It is necessary to note that if the formulation of the task was in a different way, the global optimum would be gained in another iteration. Since a task is to find the global optima, the whole subspace of potential solutions must be search for and it is not possible to shorten the computation.

[2]It was not necessary to compute the whole task. Some variables have been fixed to prescribed values, here 2 out of 8 variables, and the algorithm has been run with this restriction.

Tab. 5: A comparison of results for the 25-bar truss discrete case from literature and the present work

Variable	Units	this paper	Kripka	Lemonge & Barbosa	Li & Liu	Wu & Chow	Coello	Rajeev & Krishnamoorthy
		B & B	SA	GA	PSO	GA	GA	GA
		2011	2004	2004	2009	1995	1994	1992
A_1	in^2	0.1	0.1	0.1	0.1	0.1	1.5	0.1
A_2	in^2	0.4	0.4	0.3	0.3	0.5	0.7	1.8
A_3	in^2	3.4	3.4	3.4	3.4	3.4	3.4	2.3
A_4	in^2	0.1	0.1	0.1	0.1	0.1	0.7	0.2
A_5	in^2	2.2	2.2	2.1	2.1	1.5	0.4	0.1
A_6	in^2	1	1	1	1	0.9	0.7	0.8
A_7	in^2	0.4	0.4	0.5	0.5	0.6	1.5	1.8
A_8	in^2	3.4	3.4	3.4	3.4	3.4	3.2	3
m	lb	**484.33**	**484.33**	484.85	484.85	486.29	539.78	546.01
$\max \lvert\sigma_i\rvert$	ksi	6.20	6.20	6.11	6.11	6.01	6.66	6.77
$\max \lvert w_j\rvert$	in	0.35	0.35	0.35	0.35	0.35	0.34	0.35
σ_{lim}	ksi	40	40	40	40	40	40	40
w_{lim}	in	0.35	0.35	0.35	0.35	0.35	0.35	0.35

Tab. 5 shows the optimum gained with the branch and bound method as well as optima obtained by heuristic algorithms found in literature. The obtained result by the branch and bound method is identical to the solution presented by author Kripka. He used the Simulated Annealing method. However, he did not search the whole subspace of possible solutions so he could not be sure that the obtained optimum is the global one. It can be seen that the results of the discrete and continuous case of the optimization problem are near to each other, see Tab. 4 and Tab. 5. Therefore, the solution is potentially correct. However, we can be sure that we have found the global optimum because we have systematically explored the whole space where the global optimum is located.

The branch and bound method is suitable for bigger structures. It does not enumerate all potential solutions of the optimization problem contrary to the enumeration method. The space is restricted to the subspace between the lower and the upper bound where the global optimum is located. The lower bound is obtained e.g. with some continuous optimization method. The constrained nonlinear programming using sequence of parameterized unconstrained optimization was used in this paper. The upper bound can be gained with some heuristic method which is fast and quite effective. The more accurate the value is, the efficient the branch and bound method is. The task will not be branched to so many subproblems.

Global optima for computational demanding tasks such as the 25-bar truss problem have not been published yet to the best authors' knowledge. We hope that by publishing the algorithm as well as the value of the global optimum we will introduce a standard of quality that will help to improve new optimization methods.

Acknowledgments

This work was supported by the Grant Agency of the Czech Technical University in Prague, grants No. SGS11/021/OHK1/1T/11 and No. SGS12/027/OHK1/1T/11 and the Czech Science Foundation GACR, grant No. P105/12/1146.

References

Arora, J. S. (2002), Methods for Discrete Variable Structural Optimization. In: *Recent Advantages in Optimal Structural Design* (S. A. Burns ed.). American Society of Civil Engineers, Reston (Virginia). Chapter 1, pp 1-40.

Bendsoe, M. P., Sigmund O. (2003), *Topology Optimization: Theory, Methods and Applications*, Springer.

Coello, C. A. C. (1994), Discrete Optimization of Trusses Using Genetic Algorithms. *EXPERSYS-94: Expert Systems Applications and Artificial Intelligence* (J. G. Chen et al. eds.), I.I.T.T. International, pp. 331-336.

Dréo, J., Pétrowski, A., Siarry, P. et al. (2005), *Metaheuristics for Hard Optimization: Methods and Case Studies*, Springer.

Eiben, A. E., Smith, J. E. (2003), *Introduction to Evolutionary Computing*, Springer.

Fox, R. L., Schmit, L. A. Jr. (1966), Advances in the Integrated Approach to Structural Synthesis. *Journal of Spacecraft and Rockets*, Vol 3, No.6, pp 858-866.

Kripka, M. (2004), Discrete Optimization of Trusses by Simulated Annealing. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, Vol 26, No.2, pp 170-173.

Land, A. H., Doig A. G. (1960), An Automatic Method of Solving Discrete Programming Problems. *Econometrica*, Vol 28, No. 3, pp. 497-520.

Lee J., Hajela P. (2001), Application of classifier systems in improving response surface based approximations for design optimization. In: *Computers & Structures*, Vol 79, Is. 3, pp 333-344.

Lemonge, A. C. C., Barbosa, H. J. C. (2004), An adaptive penalty scheme for genetic algorithms in structural optimization. *International Journal for Numerical Methods in Engineering*, Vol 59, No. 5, pp. 703-736.

Li, L. J., Huang, Z. B. and Liu, F. (2009), A heuristic particle swarm optimization method for truss structures with discrete variables. *Computers & Structures*, Vol 87, No. 7-8, pp. 435-443.

Luczczek, P. (2009), *Parallel Programming in MATLAB*. [online], last revision 20/07/09 [cit. 20/07/2009]. Available from `http://web.eecs.utk.edu/ luszczek/pubs/parallelmatlab.pdf`

Naylor, P. J. (2009), *Profiling, Optimization and Acceleration of MATLAB code*. [online], last revision 20/11/09 [cit. 20/11/2009]. Available from `http://www.enm.bris.ac.uk/staff/pjn/teaching/HPC/opt-notes.pdf`

Perez, R. E., Behdinan, K. (2007), Particle Swarm Optimization in Structural Design. In: *Swarm Intelligence, Focus on Ant and Particle Swarm Optimization* (F. T. S. Chan and M. K. Tiwari eds.). Itech Education and Publishing, Vienna, (Austria), pp 373-394.

Pospíšilová A. (2010). *Analysis of sizing optimization benchmarks* (in Czech), Bachelor Thesis, CTU in Prague, FCE, Department of mechanics, Prague.

Rajeev, S., Krishnamoorthy, C. S. (1992), Discrete Optimization of Structures Using Genetic Algorithms. *Journal of Structural Engineering*, Vol 118, No. 5, pp 1233-1250.

Shewchuk, J. R. (1994), *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*, School of Computer Science, Carnegie Mellon Univ., Pittsburgh, [online].

The MathWorks (2011), *Constrained Nonlinear Optimization Algorithms: Optimization Algorithms and Examples (Optimization Toolbox)*. [online], last revision 02/10/2011 [cit. 02/10/2011]. Available from `http://www.mathworks.com/help/toolbox/optim/ug/brnoxzl.html`

The MathWorks (2011), *Executing Simultaneously on Multiple Data Sets: Single Program Multiple Data (spmd) (Parallel Computing Toolbox)*. [online], last revision 02/10/2011 [cit. 02/10/2011]. Available from `http://www.mathworks.com/help/toolbox/distcomp/brukbno-2.html`

The MathWorks (2011), *Find minimum of constrained nonlinear multivariable function - MATLAB*. [online], last revision 02/10/2011 [cit. 02/10/2011]. Available from `http://www.mathworks.com/help/toolbox/optim/ug/fmincon.html`

The MathWorks (2011), *Getting Started with parfor: Parallel for-Loops (parfor) (Parallel Computing Toolbox)*. [online], last revision 02/10/2011 [cit. 02/10/2011]. Available from `http://www.mathworks.com/help/toolbox/distcomp/brb2x2l-1.html`

Wu, S.-J., Chow, P.-T. (1995), Steady-State Genetic Algorithms for Discrete Optimization of Trusses. *Computers & Structures*, Vol 56, No.6, pp 979-991.