Effects of modeling decisions on the efficiency of solving nonlinear optimization problems

Summary of Ph.D. thesis
PhD School in Computer Science

Elvira Dobjánné Antal

Supervisors:
Dr. Tibor Csendes and Dr. Tamás Vinkó

Department of Computational Optimization
University of Szeged
2017
Szeged

INTRODUCTION

Formalization decisions in mathematical programming could significantly influence the complexity of the problem, and so the efficiency of the applied solver methods. Beyond individual intuition, reformulation of an optimization problem could be done also by automatic methods. For example, mathematicians in the 1970s were interested in the possibility of symbolic preprocessing of LP problems for the simplex method. Those early results today are utilized in the AMPL processor as an automatic "presolving" mechanism.

Some research concentrate on advantageous reformulation of (mixed-)integer programs, however these transformations usually go hand in hand with relaxation of some constraints and with the increase of the dimension of the problem. But automatically producible equivalent transformations of nonlinear optimization problems are relatively under-examined. The first part of my dissertation concentrates on that field.

In light of theoretical results in the literature it was supposed that modern computer algebra systems are capable to host a symbolic (thus reliable) application for producing equivalent transformations of nonlinear optimization problems. That would enable integrating and testing former algorithms and would inspire further refinements. A Maple program with this aims was realized, which was executing nonlinear coordinate transformations on unconstrained nonlinear optimization problems. An extensive computational test has been completed to narrow the critical fields of such an application, and I concluded that some undocumented features of Maple put obstacles in the way of development.

I studied possible commercial and open source alternatives and I found Mathematica to be the best basis for further development because of its flexibility and constantly renewed functionality. So I reimplemented the former program in Mathematica, extended it by new features, and I prove empirically, that our symbolic method are useful as an automatic presolving phase of a numerical global optimization method.

I had new theoretical results for generalize nonlinear coordinate transformations by describing possible parallel substitutions and handling constrained nonlinear optimization problems. These results were

published in a journal article with impact factor [2] and in a further peer-reviewed journal article [1].

Second part of my PhD thesis examines modeling questions in connection with a given problem, namely computing max-min fair bandwidth allocation in distributed content-sharing systems. I gave a new, exact mathematical programming formulation and algorithm for the problem, and I proved the correctness of the new algorithm theoretically. The algorithm was implemented in AMPL, and it was compared by numerical experiments to the previously proposed method for the same problem. Results are favorable in several aspects. These results were also accepted for publication in a journal article with impact factor [3].

During the implementation, in connection with the theoretical results, numerous modeling idea arose, so a whole chapter analyze their impact on efficiency of the solver. An extensive computational study was transacted for compare twelve model variants of the max-min fair bandwidth allocation problem. The test involved twenty-seven test case and two professional solvers. The results were summarized in a manuscript, submitted for publication [7].

NONLINEAR COORDINATE TRANSFORMATIONS FOR SIMPLIFICATION OF NONLINEAR OPTIMIZATION PROBLEMS

Background

We concentrate on unconstrained nonlinear optimization problems of the form

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}), \tag{1}$$

where $f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ is a smooth function given by a formula, i.e. a symbolic expression. "Expression" denotes a well-formed, finite combination of symbols (constants, variables, operators, function names and brackets). In popular computer algebra systems, as in Maple [8] and in Mathematica [11], every expression is realized by a nested list of pointers, which is in fact a directed acyclic graph, DAG [10].

The simplifier method aims to recognize whether (1) could be transformed into an equivalent formulation, which is better in the following sense: the new formulation has fewer arithmetic operations to execute during evaluation, the dimension of the problem is less, or it is simpler to solve for another reason. Equivalent means here, that a bijective transformation can be given between the optima of the original and those of the transformed problem.

Problem (1) is easy to solve when f is unimodal, that is, f has a single region of attraction and so there is only one local minimizer point (and no local maximizer point) in the given set of feasibility $X \subseteq \mathbb{R}^n$. One may recognize this property intuitively on one-dimensional functions, but in higher dimensions the decision is certainly not trivial in general.

The implicit unimodality of a function can be formulated by variable transformations according to the result of Csendes and Rapcsák:

THEOREM 1 [6] The continuous function $f(\mathbf{x})$ is unimodal in the n-dimensional real space if and only if there exists a homeomorph variable transformation $\mathbf{y} = h(\mathbf{x})$ such that $f(\mathbf{x}) = f(h^{-1}(\mathbf{y})) = \mathbf{y}^T\mathbf{y} + c$, where c is a real constant, and the origin is in the range S of $h(\mathbf{x})$.

The following theorem states sufficient conditions for substitutions that simplify the objective function:

THEOREM 2 [6] If $h(\mathbf{x})$ is smooth and strictly monotonic in x_i, then the corresponding transformation simplifies the function in the sense that each occurrence of $h(\mathbf{x})$ in the expression of $f(\mathbf{x})$ is padded by a variable in the transformed function $g(\mathbf{y})$, while every local minimizer (or maximizer) point of $f(\mathbf{x})$ is transformed to a local minimizer (maximizer) point of the function $g(\mathbf{y})$.

Furthermore, if another condition holds for the range of the substitution, then it is a bijection between the solutions of the original and the transformed objectives:

THEOREM 3 [6] If $h(\mathbf{x})$ is smooth, strictly monotonic as a function of x_i, and its range is equal to $\mathbb{R}$, then for every local minimizer (or maximizer) point $\mathbf{y}^*$ of the transformed function $g(\mathbf{y})$ there exists an x^* such that $\mathbf{y}^*$ is the transform of $\mathbf{x}^*$, and $\mathbf{x}^*$ is a local minimizer (maximizer) point of $f(\mathbf{x})$.

In the sense of the last theorem, an objective function $g(\mathbf{y})$ is equivalent to $f(\mathbf{x})$ in (1), if we get $g(\mathbf{y})$ by the following transformation:

- apply a substitution for $f(\mathbf{x})$:

$$y_i := h(\mathbf{x}), \quad 1 \leqslant i \leqslant n,$$

  where $h(\mathbf{x})$ is a continuous function with range $\mathbb{R}$, and it is strictly monotonic as a function of at least one variable x_i,

- rename the remaining variables:

$$y_j := x_j, \quad j = 1, \ldots, i-1, i+1, \ldots, n, \quad \text{and}$$

- omit the variables y_i without presence in the evolving objective function.

We said, that $h(\mathbf{x})$ *covers variable* x_i in $f(\mathbf{x})$, if $h(\mathbf{x})$ characterizes all occurrences of x_i in $f(\mathbf{x})$, that is, x_i could be removed totally from the optimization problem by substituting $h(\mathbf{x})$ by y_i.

The term *appropriate substitution* will refer to an $y_i = h(\mathbf{x})$ substitution, where

- $h(\mathbf{x})$ satisfies the criteria being smooth, monotonic in at least one variable x_i, and its range is equal to $\mathbb{R}$,

- $h(\mathbf{x})$ covers at least one variable x_i, and
- $y_i = h(\mathbf{x})$ is not a simple renaming, that is, $h(\mathbf{x}) \neq x_i$, $i = 1, \ldots, n$.

After applying a transformation with an appropriate substitution $y_i = h(\mathbf{x})$, $\mathbf{y}$ has at most the same dimension as $\mathbf{x}$. Redundant variables can be eliminated, if $h(\mathbf{x})$ covers two or more variables. In other words, we have the possibility to recognize whether the model can be formalized with a smaller set of variables. Recognition of redundant variables is beneficial and it is usually by far not trivial. In this way the outcome of the transformation sequence could be favorable even if a unimodal problem form could not be reached.

For example, consider the minimization of $f(x_1, x_2) = (x_1 + x_2)^2$. It is equivalent to minimize $g(y_1) = y_1^2$, and the optimal values of the original variables x_1 and x_2 can be set by the symbolic equation $y_1 = x_1 + x_2$, which is an appropriate substitution. In fact, we can handle an infinite number of global optimum points in this way, which is impossible for any numerical solver.

One of the main goals of the simplifier method is to find appropriate substitutions which would eliminate variables. Csendes and Rapcsák with their Assertion 2 suggest to compute the partial derivatives $\partial f(\mathbf{x})/\partial x_i$, factorize them, and search for appropriate substitutions in the factors. The following two theoretical statements provide additional hints on how to identify eliminable variables:

ASSERTION 1 [6] If a variable x_i appears everywhere in the expression of a smooth function $f(\mathbf{x})$ in a term $h(\mathbf{x})$, then the partial derivative $\partial f(\mathbf{x})/\partial x_i$ can be written in the form $(\partial h(\mathbf{x})/\partial x_i)\, p(\mathbf{x})$, where $p(\mathbf{x})$ is continuously differentiable.

ASSERTION 2 [6] If the variables x_i and x_j appear everywhere in the expression of a smooth function $f(\mathbf{x})$ in a term $h(\mathbf{x})$, then the partial derivatives $\partial f(\mathbf{x})/\partial x_i$ and $\partial f(\mathbf{x})/\partial x_j$ can be factorized in the forms $(\partial h(\mathbf{x})/\partial x_i)\, p(\mathbf{x})$ and $\left(\partial h(\mathbf{x})/\partial x_j\right) q(\mathbf{x})$, respectively, and $p(\mathbf{x}) = q(\mathbf{x})$.

If $\partial f(\mathbf{x})/\partial x_i$ is not factorisable, then any appropriate substitution that is monotonic as a function of x_i is linear as a function of x_i.

These theoretical statements are the basis for the constructive symbolic algorithm, the Maple and Mathematica implementations of which were realized.

The assertions suggest to compute the partial derivatives $\partial f(\mathbf{x})/\partial x_i$, factorize them, and search for appropriate substitutions in the factors.

Obviously, finding the appropriate $(\partial h(\mathbf{x})/\partial x_i)\, p(\mathbf{x})$ factorization is not necessarily easy, as many other factorizations can exist beside the one mentioned in Assertion 1. Yet a canonical form can be derived for a wide class of functions, e.g. for a polynomial $f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$ which have n roots $(x_1, \ldots, x_n)$ a standard factorization $a_n(x - x_1)(x - x_2)\cdots(x - x_n)$ always exists.

Even if $\partial h(\mathbf{x})/\partial x_i$ can be determined, it may be difficult to find a good transformation function $h(\mathbf{x})$. The mentioned conditions are sufficient, but not necessary conditions for simplification transformations.

My contributions

I. 1 *Implementation of the automatic symbolic simplification method in Maple and Mathematica.*

The method for simplifying unconstrained nonlinear optimization problems, based on the referred theoretical results, was implemented in Maple and Mathematica environments. The program can automatically recognize helpful transformations of the mathematical model and detect implicit redundancy in the objective function [2, 1]. Redundant variables can be eliminated, we have the possibility to recognize whether the model can be formalized with a smaller set of variables.

In Section 3.4 of the dissertation I discussed the main problems arisen in connection with the first implementation in Maple. Maple and Mathematica based implementations were compared from the produced substitutions point of view. In this part, I could rely especially on my custom made problems designed especially to test the capabilities of symbolic simplification algorithms. It turned out that Mathematica is more favorable as the environment of our simplification method due to its highly customizable substitution routine and better interval arithmetic capabilities.

I. 2 *Computational test completed on standard global optimization test problems and as a presolving phase of a numerical solver.*

In Section 3.4 of the dissertation 45 well-known global optimization test problems were examined, and our Mathematica program offered equivalent transformations for 8 cases [1], what is 18% of this test set. It is impressive if we consider that no other method is known for suggesting similar automatic reformulations for those problems.

All substitutions produced by the Mathematica version of the automatic simplification method are correct. For decide, whether the produced substitutions are useful or negligible, the ability of using equivalent transformations for nonlinear optimization problems as an automatic presolving phase of a numerical global optimization method were examined. More specifically, tests with a numerical solver, namely GLOBAL, were performed to check the usefulness of the produced transformations from the running times and function evaluation numbers point of view [1]. The computational results obtained for standard global optimization test problems and for other artificially constructed instances show that the produced substitutions can improve the performance of this multi-start solver. The transformations impact running times and function evaluation numbers in a favorable manner. The average relative improvement in the running times and also in the number of function evaluations on the whole test set is 32% due to the transformations.

I. 3 *Generalization of theory to describe possible parallel substitutions and constrained NLPs.*

I generalized nonlinear coordinate transformations of Csendes and Rapcsák [1] in two directions. The results are presented in Section 3.5 of the dissertation. At first, I give sufficient conditions for more complicated equivalent transformations by describing possible parallel substitutions. Secondly, I extend the scope of the method to constrained nonlinear optimization problems.

The original nonlinear optimization problem that will be simpli-

fied automatically now can include constraints, too:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ c_i(x) \quad & = 0 \\ c_j(x) \quad & \leqslant 0, \end{aligned} \tag{2}$$

where $f(x), c_i(x), c_j(x) : \mathbb{R}^n \to \mathbb{R}$ are smooth real functions, given by a formula, and $i = 1, \ldots, p_1$ and $j = p_1 + 1, \ldots, p_2$ are integer indexes.

The transformed constrained optimization problem will be

$$\begin{aligned} \min_{y \in \mathbb{R}^m} \quad & g(y) \\ d_i(y) \quad & = 0 \\ d_j(y) \quad & \leqslant 0, \end{aligned} \tag{3}$$

where $g(y) : \mathbb{R}^m \to \mathbb{R}$ is the transformed form of $f(x)$, and $d_i(y), d_j(y) : \mathbb{R}^m \to \mathbb{R}$ are again smooth real functions, the transformations of the constraints $c_i(x), c_j(x)$, $i = 1, \ldots, p_1$ and $j = p_1 + 1, \ldots, p_2$.

Denote by X the set of variable symbols in the objective function $f(x)$ and in the constraint functions $c_k(x)$, $k = 1, \ldots, p_2$. Y will be the set of variable symbols in the transformed functions $g(y)$, and $d_k(y)$, $k = 1, \ldots, p_2$. Remark, that dimension increase is not allowed for the transformation steps, so $m \leqslant n$ and $|Y| \leqslant |X|$. At the beginning of the algorithm, $Y := X$.

Denote the set of the expressions on the left-hand side of the original constraints by C:

$$C := \{c_k(x) : \mathbb{R}^n \to \mathbb{R}\,,\ k = 1, \ldots, p_2\}.$$

Denote by F the expression set of all subexpressions (all well-formed expressions that are part of the original expressions) of $f(x)$ and $c_k(x) \in C$.

The crucial part of our algorithm is the *transformation step*. If an $H \subset F$ expression set covers a $V \subseteq X$ variable set (that is, none

of $v \in V$ happens out of H in the original expression of $f(\mathbf{x})$ or $c_k(\mathbf{x}) \in C$), and $|H| \leqslant |V|$, then apply every substitutions, related to H, to $f(\mathbf{x})$ and C as follows. Substitute a new variable y_i in place of $h_i(\mathbf{x})$ for all $h_i(\mathbf{x}) \in H$ in $f(\mathbf{x})$ and also in every $c_k(\mathbf{x}) \in C$. Furthermore, let us update the variable set of the transformed problem: $Y := (Y \cup y_i) \setminus V$.

Abban a speciális esetben, ha $|H| = |V| = 1$ és $p_1 = p_2 = 0$, visszakapjuk Csendes és Rapcsák [6] algoritmusát a feltétel nélküli esetre. This will be referred to as a transformation step (corresponding to H). The special case $|H| = |V| = 1$, $p_1 = p_2 = 0$ belongs to the algorithm given by Csendes and Rapcsák [6] for the unconstrained case.

Further on, the notation $\mathbf{y} := H(\mathbf{x})$ will be used as an abbreviation for the following: $y_i := h_i(\mathbf{x})$, $i = 1, \ldots, |H|$

THEOREM 4 If H is proper and all $h_i(\mathbf{x}) \in H$ expressions are smooth and strictly monotonic as a function of every variable $v \in V \subseteq X$, the cardinality of H is less than or equal to the cardinality of V, and the domain of $h_i(\mathbf{x})$ is equal to $\mathbb{R}$ for all $h_i(\mathbf{x}) \in H$, then the transformation step corresponding to H simplifies the original problem in such a way that every local minimizer (maximizer) point $\mathbf{x}^*$ of the original problem is transformed to a local minimizer (maximizer) point $\mathbf{y}^*$ of the transformed problem.

THEOREM 5 If H is proper, and all $h_i(\mathbf{x}) \in H$ expressions are smooth, strictly monotonic as a function of every variable $v \in V \subseteq X$, the cardinality of H is less than or equal to the cardinality of V, and the domain and range of $h_i(\mathbf{x})$ are equal to $\mathbb{R}$ for all $h_i(\mathbf{x}) \in H$, then the transformation step corresponding to H simplifies the original problem in such a way that every local minimizer (maximizer) point $\mathbf{y}^*$ of the transformed problem can be transformed back to a local minimizer (maximizer) point $\mathbf{x}^*$ of the original problem.

MAX-MIN FAIR BANDWIDTH ALLOCATION IN BITTORRENT NETWORKS

Background

I have studied the max-min fair bandwidth allocation problem of BitTorrent communities at inter-swarm level, which is one of the most complex levels due to the behavior of users, as the model handles that most of the users are participating in uploading and downloading multiple contents at the same time.

For modeling the state of a BitTorrent community at a certain instant, we use the graph-theoretical model introduced in [5], which can be summarized as follows. A BitTorrent community consists of a set I of users and a set T of torrents. Each user $i \in I$ has upload bandwidth μ_i and download bandwidth δ_i. The flow network representation of a BitTorrent community is $G = (\{U, L, D\}, E, f, c)$, which is a directed, bipartite, weighted graph, where

$U = \{ u_i \mid i \in I\}$: the *upload nodes* of G, where u_i represents the upload (seeding or leeching) potential of user i;

$D = \{ d_i \mid i \in I\}$: the *download nodes* of G, where d_i represents the download (leeching) potential of user i;

$L = \left\{ l_i^t \mid i \in I,\, t \in T \right\}$: the *leeching nodes* of G, where the presence of l_i^t, called *leeching session*, denotes that user i leeches actually torrent t;

E: the set of edges $E = E_U \cup E_D$, where $E_U = \bigcup_{i,j,t}(u_i, l_j^t)$ is the set of *upload edges*, and $E_D = \bigcup_{j,t}(l_j^t, d_j)$ is the set of *download edges*;

$c : U \cup L \cup D \to \mathbb{N}$: the *capacity function* represents the bandwidth constraints of the peers:

$$c(u_i) = \mu_i,\ c(d_i) = \delta_i,\ c(l_i^t) = \infty;$$

$f : E \to \mathbb{R}^+$: the *flow function* represents the bandwidth allocation on the edges satisfying the flow conservation property:

$$\sum_{u_i \in U} f(u_i, l_j^t) = f(l_j^t, d_j) \qquad (\forall l_j^t \in L),$$

as well as the *capacity constraints*:

$$\sum_{t,j} f(u_i, l_j^t) \leqslant \mu_i \qquad \forall (u_i, l_j^t) \in E_U,$$

$$\sum_{t} f(l_j^t, d_j) \leqslant \delta_j \qquad \forall (l_j^t, d_j) \in E_D.$$

Using this flow network model, a bandwidth allocation is *max-min fair* if the flow value $f(l_j^t, d_j)$ on a download edge (l_j^t, d_j) can only be increased by decreasing the flow value $f(l_{j'}^{t'}, d_{j'})$ on another download edge $(l_{j'}^{t'}, d_{j'})$ for which $f(l_{j'}^{t'}, d_{j'}) < f(l_j^t, d_j)$.

As it was already stated in [5], the problem is formulated on continuous and convex sets, hence the max-min fair allocation uniquely exists [4, 9].

It was shown by Capotă *et al*. [5] that using the standard BitTorrent protocol's bandwidth allocation, the average performance of a BitTorrent community is suboptimal in terms of max-min fairness. This fairness measure corresponds to the case of video-streaming service – an emerging application of peer-to-peer networks.

Our algorithm [3] is a refinement of the work of Capotă et al. It is an adapted version of the general Max-Min Programming Algorithm summarized by Radunović and Le Boudec [9], as it computes the max-min fair weights of the download edges with an iterative manner, by fixing the coordinates with the smallest unfixed weight in every iteration.

My contributions

II. 1 *Exact mathematical programming formulation and algorithm with proof of correctness.*

My main contributions are to replace the complicated sieve method of Capotă et al. by an exact mathematical programming formulation, and to prove the correctness of our *mMaxMin* algorithm based on that formulation. The proposed algorithmic approach with the proof of correctness is detailed in a journal article with impact factor [3], and also in Section 5.4 of the dissertation.

The following meaningful lemma and theorem does not need further preparations.

LEMMA 2 For every iteration k of *mMaxMin*

$$\sigma_k = \sigma$$

holds, where σ denotes a constant, the maximum throughput of the network such that

$$\forall (l_j^t, d_j) \in E_D \; : \; f(l_j^t, d_j) \geqslant \phi.$$

THEOREM 6 The *mMaxMin* terminates in finite iterations, and guarantees the max-min fairness property for every download edge.

II. 2 *AMPL implementation and numerical tests.*

The correct algorithm was implemented in AMPL with the application of my theoretical results and some further reformulation techniques. Section 5.5 presents numerical experiments on large problem instances derived from snapshots of the BitSoup.org community, including comparison with the previously proposed method for the same problem. Our observations show that our formulation helps the Gurobi solver to achieve shorter running times, and *MaxMin-r* can be faster than the earlier proposed *MM* algorithm on larger problem instances. Moreover, the results from the first iterations of *MaxMin-r* could be used as a very good approximation for the max-min fair allocation. This approximation, which is a feasible solution, can be achieved in a fraction of the time of the adequate precession of *MM*. For example, the first iteration of *MaxMin-r* took 46 seconds for the test case containing 23 670 nodes and 7 326 edges, compared to the more than eight-hour running time for the first 910 iterations of *MM* with adequate solution quality.

II. 3 *Analyzing impact of modeling techniques.*

Based on our manuscript submitted for publication [7], in Chapter 6 of the dissertation I analyzed the effects of using several modeling techniques in the optimization problem for computing max-min fair bandwidth allocation in a BitTorrent community. The problem includes the solution of large linear and nonlinear mixed-integer programs. An extensive computational study was

transacted including every combination of twelve model variants, twenty-seven test case and two professional solvers.

The performed experiments hand on several interesting results. First of all, there was no model variant with quickest solution for every case. It was pointed out, that the tested modern solvers are capable to solve efficiently the bilinear form of a problem. Gurobi and MOSEK produced better running times for the smaller bilinear problems, in comparison with the equivalent linear forms with higher dimensions, which were produced by applying McCormick envelopes. An LP presolving technique, also implemented in AMPL, was prominent, as it induced 10% decrease of running times for the test cases on average.

REFERENCES

[1] ANTAL, E. – CSENDES, T.: Nonlinear symbolic transformations for simplifying optimization problems. In *Acta Cybernetica*, Vol 22 (2016) No 4, 715–733. pp.

[2] ANTAL, E. – CSENDES, T. – VIRÁGH, J.: Nonlinear transformations for the simplification of unconstrained nonlinear optimization problems. In *Central European Journal of Operations Research (CEJOR)*, Vol 21 (2013) No 4, 665–684. pp.

[3] ANTAL, E. – VINKÓ, T.: Modeling max–min fair bandwidth allocation in BitTorrent communities. In *Computational Optimization and Applications*, 2016. 18 pp., `http://dx.doi.org/10.1007/s10589-016-9866-5`. Accepted for publication.

[4] BERTSEKAS, D. pp. – GALLAGER, R. G.: *Data Networks*. 2nd. kiad. Englewood Cliffs, NJ, USA, Prentice Hall, 1992. ISBN 0132009161.

[5] CAPOTĂ, M. – ANDRADE, N. – VINKÓ, T. – SANTOS, F. – POUWELSE, J. – EPEMA, D.: Inter-swarm resource allocation in BitTorrent communities. In *Proceedings of IEEE International Conference on Peer-to-Peer Computing (P2P 2011)* (conference material). 2011, 300–309. pp.

[6] CSENDES, T. – RAPCSÁK, T.: Nonlinear coordinate transformations for unconstrained optimization I. Basic transformations. In *Journal of Global Optimization*, Vol 3 (1993) No 2, 213–221. pp.

[7] DOBJÁNNÉ ANTAL E. – VINKÓ T.: Egy nemlineáris vegyes-egészértékű optimalizálási feladat különféle modelljeinek komparatív elemzése (in hungarian). 2016. `http://www.inf.u-szeged.hu/~antale/en/research/DAntal_MIPmodels2016.pdf`. Submitted for publication.

[8] HECK, A.: Internal data representation and substitution. In *Introduction to Maple*. New York, NY, Springer New York, 2003, 153–174. pp. ISBN 978-1-4613-0023-6.

[9] RADUNOVIĆ, B. – LE BOUDEC, J.-Y.: A unified framework for max-min and min-max fairness with applications. In *IEEE/ACM Transactions on Networking*, Vol 15 (2007) No 5, 1073–1083. pp.

[10] SCHICHL, H. – NEUMAIER, A.: Interval analysis on directed acyclic graphs for global optimization. In *Journal of Global Optimization*, Vol 33 (2005) No 4, 541–562. pp.

[11] Wolfram Language & System Documentation center. Wolfram Language tutorial: Basic internal architecture. `http://reference.wolfram.com/language/tutorial/BasicInternalArchitecture.html`. [2017/01/12].