Using the Second-Order Information in Pareto-set Computations of a Multi-criteria Problem

Bartłomiej Jacek Kubica and Adam Woźniak

Institute of Control and Computation Engineering,
Warsaw University of Technology, Poland
bkubica@elka.pw.edu.pl, A.Wozniak@ia.pw.edu.pl

Abstract. The paper presents an extension of a previously developed interval method for solving multi-criteria problems [13]. The idea is to use second order information (i.e., Hesse matrices of criteria and constraints) in a way analogous to global optimization (see e.g. [6], [9]). Preliminary numerical results are presented and parallelization of the algorithm is considered.

Keywords: Pareto-front, Pareto-set, multi-criteria analysis, interval computations, second-order optimality conditions.

1 Introduction

We consider seeking the Pareto-set of the following problem:

$$\begin{aligned}
&\min_x q_k(x) \qquad k = 1, \ldots, N \ , \\
&\text{s.t.} \\
&g_j(x) \le 0 \qquad j = 1, \ldots, m \ , \\
&x_i \in [\underline{x}_i, \overline{x}_i] \qquad i = 1, \ldots, n \ .
\end{aligned} \tag{1}$$

Definition 1. *A feasible point x is* Pareto-optimal *(nondominated), if there exists no other feasible point x' such that:*

$$\begin{aligned}
(\forall k) \quad & q_k(y) \le q_k(x) \ \textit{and} \\
(\exists i) \quad & q_i(y) < q_i(x) \ .
\end{aligned}$$

The set $P \subset \mathbb{R}^n$ of all Pareto-optimal points (Pareto-points) is called the Pareto-set.

Definition 2. *The Pareto-front is the image of the Pareto-set, i.e., the set of criteria values for all nondominated points.*

In the sequel one more definition will be needed.

Definition 3. *A point y* dominates *a set B, iff $D(y) \cap B = \emptyset$ and similarly a set B'* dominates *a set B, iff $(\forall y \in B') D(y) \cap B = \emptyset$.*

The interpretation of the definitions is straightforward. A feasible point is Pareto-optimal if there is no other feasible point that would reduce some criterion without causing a simultaneous increase in at least one other criterion. Pareto-front is the image of Pareto-set in criterion space and D is the cone of domination in this space.

Interval methods allow to solve the problem of approximating the Pareto-set, using the branch-and-bound principle. Starting from the initial box and bisecting the boxes subsequently, we can quickly discard dominated boxes, enclosing the Pareto-set and Pareto-front with sets of boxes in decision and criteria spaces.

To discard or narrow boxes the algorithms use the following tools:

– checking if the box is non-dominated by other boxes (i.e., it may contain non-dominated points),
– set inversion of boxes from the criteria space to the decision space,
– the monotonicity test adapted to multi-criteria case (this test uses the first-order information).

No currently used interval algorithm ([3], [13], [16]) for computing the Pareto set uses the second-order information; gradients of the criteria and constraints are used, but not the Hesse matrices. The method of Toth and Fernandez [4] allows it by reducing the problem of Pareto-front seeking to repeated global optimization, but the approach can be applied to bi-criteria problems only.

Our idea is to extend the method proposed in [13] by using Hesse matrices of criteria and constraints in a way similar to well-known global optimization algorithms (see e.g. [6]), i.e., by solving the system of second order optimality conditions (in this case: Pareto-optimality conditions).

2 Generic Algorithm

In previous papers we developed an algorithm to seek the Pareto-set. It subdivides the criteria space in a branch-and-bound manner and inverts each of the obtained sets using a version of the SIVIA (Set Inversion Via Interval Analysis) procedure [8]. This version uses some additional tools (like the componentwise Newton operator) to speedup the computations.

The algorithm is expressed by the following pseudocode described with more details in previous papers ([13], [15]).

```
compute_Pareto-set ($\mathtt{q}(\cdot)$, $\mathbf{x}^{(0)}$, $\varepsilon_y$, $\varepsilon_x$)
// $\mathtt{q}(\cdot)$ is the interval extension of the function
      $q(\cdot) = (q_1, \ldots, q_N)(\cdot)$
// $L$ is the list of quadruples $(\mathbf{y}, L_{\text{in}}, L_{\text{bound}}, L_{\text{unchecked}})$
// for each quadruple: $L_{\text{in}}$ is the list of interior boxes (in the decision space),
// $L_{\text{bound}}$ – the list of boundary boxes and $L_{\text{unchecked}}$ – of boxes to be checked yet
$\mathbf{y}^{(0)} = \mathtt{q}(\mathbf{x}^{(0)})$;
$L = \Big\{\big(\mathbf{y}^{(0)}, \{\}, \{\}, \{\mathbf{x}^{(0)}\}\big)\Big\}$;
while (there is a quadruple in $L$, for which $\text{wid}\,\mathbf{y} \geq \varepsilon_y$)
    take this quadruple $(\mathbf{y}, L_{\text{in}}, L_{\text{bound}}, L_{\text{unchecked}})$ from $L$;
    bisect $\mathbf{y}$ to $\mathbf{y}^{(1)}$ and $\mathbf{y}^{(2)}$;
    for $i = 1, 2$
        apply SIVIA with accuracy $\varepsilon_x$ to quadruple
            $(\mathbf{y}^{(i)}, L_{\text{in}}, L_{\text{bound}}, L_{\text{unchecked}})$;
        if (the resulting quadruple has a nonempty interior,
            i.e., $L_{\text{in}} \neq \emptyset$)
            delete quadruples that are dominated by $\overline{y}^{(i)}$;
        end if
        insert the quadruple to the end of $L$;
    end for
end while
// finish the Pareto-set computations
for each quadruple in $L$ do
    process boxes from $L_{\text{unchecked}}$ until all of them get to $L_{\text{in}}$ or $L_{\text{bound}}$;
end do;
end compute_Pareto-set
```

Obviously, both loops in the above algorithm – the `while` loop and the `for each` loop can easily be parallelized.

3 Basic Idea

Let us formulate the set of Pareto optimality conditions; similar to the John conditions set [6]. For an unconstrained problem it has the following form (notation from [10] is used):

$$\begin{aligned}
&u_1 \cdot \frac{\partial q_1(x)}{\partial x_1} + \cdots + u_N \cdot \frac{\partial q_N(x)}{\partial x_1} = 0 \ , \\
&\cdots \\
&u_1 \cdot \frac{\partial q_1(x)}{\partial x_n} + \cdots + u_N \cdot \frac{\partial q_N(x)}{\partial x_n} = 0 \ , \\
&u_1 + u_2 + \cdots + u_N = 1 \ ,
\end{aligned} \tag{2}$$

where $u_i \in [0, 1]$ $i = 1, \ldots, N$.

The above is a system of $(n+1)$ equations in $(n+N)$ variables. As the problem is supposed to have multiple criteria, clearly $N > 1$, which makes System (2) underdetermined. Solving underdetermined problems is less studied than well-determined ones (see paper [12] and references therein) and more difficult at the same time.

To consider a constrained multi-criteria problem, System (2) has to be extended slightly. In addition to multipliers u_i for all criteria $i = 1, \ldots, N$, we must have multipliers for all constraints: u_{N+j}, $j = 1, \ldots, m$.

The resulting system would take the following form:

$$\begin{aligned}
&u_1 \cdot \frac{\partial q_1(x)}{\partial x_1} + \cdots + u_N \cdot \frac{\partial q_N(x)}{\partial x_1} + \\
&\quad + u_{N+1} \cdot \frac{\partial g_1(x)}{\partial x_1} + \cdots + u_{N+m} \cdot \frac{\partial g_m(x)}{\partial x_1} = 0 \ ,
\end{aligned} \tag{3}$$

$$\begin{aligned}
&\ldots\\
&u_1 \cdot \frac{\partial q_1(x)}{\partial x_n} + \cdots + u_N \cdot \frac{\partial q_N(x)}{\partial x_n} +\\
&\quad + u_{N+1} \cdot \frac{\partial g_1(x)}{\partial x_n} + \cdots + u_{N+m} \cdot \frac{\partial g_m(x)}{\partial x_n} = 0 \ ,\\
&u_{N+1} \cdot g_1(x) = 0 \ ,\\
&\ldots\\
&u_{N+m} \cdot g_m(x) = 0 \ ,\\
&u_1 + u_2 + \cdots + u_N + u_{N+1} + \cdots + u_{N+m} = 1 \ ,
\end{aligned}$$

which is an underdetermined system of $(n + m + 1)$ equations in $(n + m + N)$ variables.

This system is used for narrowing the boxes by interval Newton operators in the SIVIA procedure. The procedure is similar to the one known from interval global optimization (see [6], [9]).

Previous experiments [12] with underdetermined equations systems, like (3) suggest that two methods are promising in solving them:

– the componentwise Newton operator [7],
– the Gauss-Seidel (GS) operator with rectangular matrix [9].

The first technique uses linearization of each equation with respect to only one of the variables at a time. Pairs equation-variable can be chosen using several heuristics. Our implementation uses the strategy of S. Herbort and D. Ratz [7] and tries to use all possible pairs subsequently.

The well-know GS operator is commonly used in interval algorithms. In our case it has to be used for a linear equations system with a rectangular matrix. This does not change much in the method: we choose one variable for reduction for each equation.

A slight adaptation of the classical GS procedure has to be done in preconditioning. We use the inverse-midpoint preconditioner, choosing a square submatrix with the Gauss elimination procedure, performed on the midpoint-matrix.

In current implementation of our algorithm we can use both versions of the Newton operator.

4 Implementation

Parallelization of the algorithm was done in a way described in [14] and [15]. This approach parallelizes the "outer loop" of the algorithm, i.e., operations on different boxes in the criteria space are done in parallel, but there is no nested parallelism on the SIVIA procedure applied to them. This allows larger grain-size, but makes us to execute costly operations on the list of sets in a critical section (deleting all dominated sets). Parallelization was obtained using POSIX threads [2] as in previous implementations [14], [15].

The program uses the C-XSC library [1] for interval operations.

5 Examples

We tested three versions of the algorithm:

– the version that uses 1st order information only; no Hesse matrices are computed nor considered,
– the version using the componentwise interval Newton operator with the S. Herbort and D. Ratz heuristic,
– the interval Gauss-Seidel operator with rectangular matrix.

Two problems to be solved were considered.

5.1 The Kim Problem

Our first example is a well-known hard problem for multi-criteria analysis [11]:

$$\begin{aligned}
\min_{x_1,x_2} \Big(q_1(x_1,x_2) &= -\big(3(1-x_1)^2 \cdot \exp(-x_1^2-(x_2+1)^2) + \\
&-10 \cdot \big(\frac{x_1}{5} - x_1^3 - x_2^5\big) \cdot \exp(-x_1^2-x_2^2) + \\
&-3\exp(-(x_1+2)^2-x_2^2) + 0.5 \cdot (2x_1+x_2)\big), \\
q_2(x_1,x_2) &= -\big(3 \cdot (1+x_2)^2 \cdot \exp(-x_2^2-(1-x_1)^2) + \\
&-10 \cdot \big(-\frac{x_2}{5} + x_2^3 + x_1^5\big) \cdot \exp(-x_2^2-x_1^2) + \\
&-3\exp(-(2-x_2)^2-x_1^2)\big)\Big), \\
x_1, x_2 &\in [-3,3] \ .
\end{aligned} \tag{4}$$

The second example is related to a practical problem.

5.2 Tuning the PI Controller

A Proportional-Integral-Derivative (PID) controller can be found in virtually all kind of control equipments. In the so-called ideal non-interacting form it is described by the following transfer function:

$$R(s) = k \cdot \big(1 + \frac{1}{Ts} + T_d s\big) \ , \tag{5}$$

with parameters k, T and T_d. The selection of these parameters, i.e., the tuning of the PID actions, is the crucial issue in the control-loop design. A large number of tuning rules has been derived in the last seventy years starting with the well-known Ziegler-Nichols algorithm.

As many other engineering design problems PID tuning is generally a multi-objective one and can be solved using multi-objective optimization techniques. We are going to present an application of the derived algorithm for PID controller tuning for a non-minimum-phase (inverse response) plant. It is well known that in

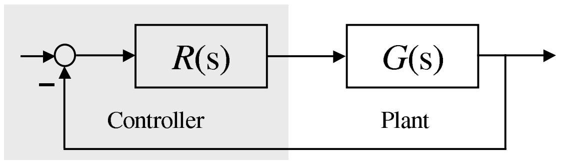

Fig. 1. Control system composed of a plant and a controller with closed-loop feedback

this case the PI controller, i.e., with proportional and integral terms, is adequate. Performance of the control loop can be measured in different way. In our example two objective functions were chosen: integrated square error (ISE) and some measure of overshoot. The ISE criterion minimized alone is insufficient, because it accepts oscillatory unit-step set-point response of the control-loop, so a second criterion has to be used. Controlled plant has inverse response, so we use as a measure of overshoot the span of closed-loop response to unit step in reference.

Applying an interval method to this problem was challenging as closed-form formulae are necessary to use them. Computing the formula for output signal of the system required computing the poles of the transition function, i.e., roots of its denominator, which is a quadratic function. The formulae are different for different signs of the discriminant $\varDelta$ of this quadratic equation. As $\mathbf{\Delta}$ is often an interval containing zero, one has to consider the interval hull of the results of all three formulae (for $\varDelta$ positive, negative and equal to 0). As the automatic differentiation toolbox of C-XSC [1] does not have such operation, changes had to be done to the library code. Moreover the formulae are quite likely to result with improper operations, like division by 0 or computing the square root of an interval containing negative values. All such cases had to be carefully implemented.

Nevertheless, using our algorithm we are able to present the Pareto-front to the control-loop designer, so that they could consider conflicting criteria simultaneously and basing, e.g. on Haimes' multi-objective trade-off analysis [5] choose the controller parameters properly.

6 Computational Experiments

Numerical experiments were performed on a computer with 16 cores, i.e., 8 Dual-Core AMD Opterons 8218 with 2.6GHz clock. The machine ran under control of a Fedora 11 Linux operating system. The solver was implemented in C++, using C-XSC 2.4.0 library for interval computations. The GCC 4.4.4 compiler was used.

The following notation is used in the tables:

- "1st order" – the version of our algorithm that uses 1st order information only; no Hesse matrices are computed nor considered,
- "Ncmp" – the version using the componentwise interval Newton operator with the S. Herbort and D. Ratz heuristic,

- "GS" – the interval Gauss-Seidel operator with rectangular matrix,
- "high acc." at the version's name means results for smaller values of ε_y and ε_x.

For the Kim problem (4) we set computational accuracies in criteria space at $\varepsilon_y = 0.2$, and in decision space at $\varepsilon_x = 10^{-3}$. As high accuracy we used $\varepsilon_y = 0.1$, and $\varepsilon_x = 10^{-4}$, respectively.

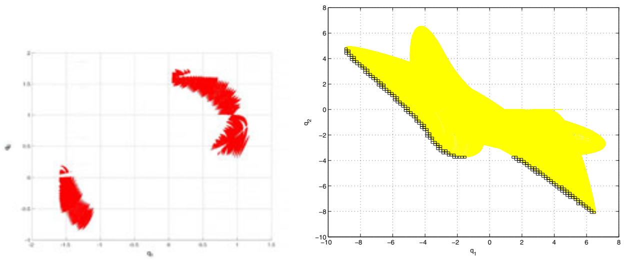

Fig. 2. Pareto-set in decision space and Pareto-front in criteria space computed for the Kim problem using 1st order information

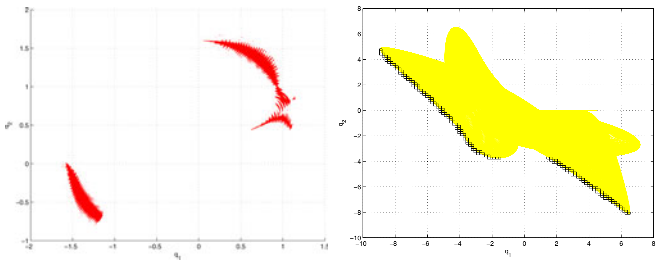

Fig. 3. Pareto-set and Pareto-front computed for the Kim problem using component-wise Newton version of the algorithm (Ncmp)

It is easy to observe by inspection that using 2nd order information results in dramatic improvement in accuracy of Pareto-set determination for the hard Kim problem.

For the PID tuning problem we used finer computational accuracies, i.e., in criteria space it was $\varepsilon_y = 0.02$, and in decision space – $\varepsilon_x = 10^{-5}$. As high accuracy we used $\varepsilon_y = 0.001$, and $\varepsilon_x = 10^{-6}$, respectively.

The obtained results allowed control designer to choose P and I parameters, i.e., k and T, resulting in very small values of rise time and inverse overshoot.

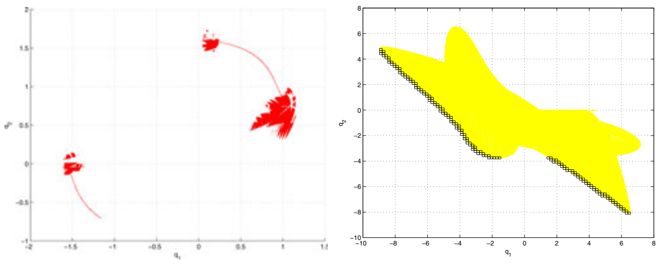

Fig. 4. Pareto-set and Pareto-front computed for the Kim problem using interval GS version of the algorithm (GS)

Table 1. Results for the Kim problem (4) and a single-threaded algorithm

	1st order	Ncmp	GS	GS (high acc.)
criteria evals.	10543390	4493434	2837044	26662794
criteria grad. evals	3906722	7642454	742694	8068624
criteria Hess. evals	0	1578964	1085072	11293078
bisecs.in crit.space	440	438	434	809
bisecs.in dec. space	956390	372062	253237	2796488
boxes deleted by monot.	17120	2234	6459	18411
boxes deleted by Newton	85716	128997	34390	274630
resulting quadruples	174	171	161	310
internal boxes	352922	101612	80782	1235138
boundary boxes	462448	94513	96066	1191551
Lebesgue measure crit.	4.84	4.76	4.48	2.16
Lebesgue measure dec.	0.76	0.24	0.19	0.13
time (sec.)	71	155	62	655

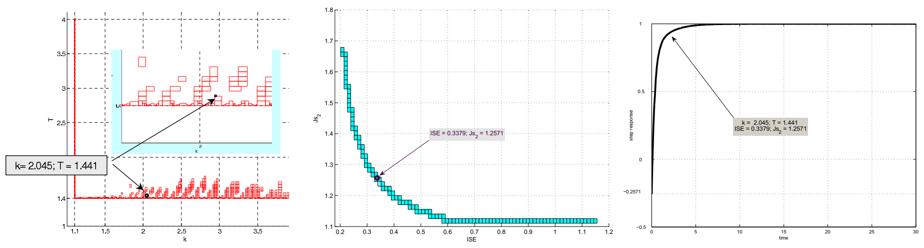

Fig. 5. Fragment of Pareto-set and Pareto-front computed for the PID tuning problem using interval GS version of the algorithm (GS) with the chosen PI controller settings and the response generated by the closed-loop system

Table 2. Results for the PID tuning problem and a single-threaded algorithm

	1st order	**Ncmp**	**GS**	**Ncmp (high acc.)**	**GS (high acc.)**
criteria evals.	396436222	228916	231500	4250430	4255408
criteria grad. evals	160959276	87394	23554	1606786	424160
criteria Hess. evals	0	154456	155306	2392690	2390932
bisecs.in crit.space	452	451	452	8261	8262
bisecs.in dec. space	40222521	20318	20532	445048	445494
boxes deleted by monot.	634	108	111	120	123
boxes deleted by Newton	170201	3819	519	54741	10159
resulting quadruples	147	145	145	3589	3589
internal boxes	23795930	9268	9273	223554	223575
boundary boxes	16250540	954	948	28910	28921
Lebesgue measure crit.	0.04	0.04	0.04	0.00192	0.00192
Lebesgue measure dec.	1.85	0.32	0.32	0.00515	0.00515
time (sec.)	4995	14	13	202	203

Table 3. Speedup for parallelized algorithms on the Kim problem

		1	2	4	6	8	10	12
1st order	time (sec.)	71	41	25	19	17	16	15
	speedup	1.0	1.73	2.84	3.74	4.18	4.44	4.73
Ncmp	time (sec.)	155	81	41	29	22	19	17
	speedup	1.0	1.91	3.78	5.34	7.05	8.16	9.12
GS	time (sec.)	62	32	16	11	9	11	10
	speedup	1.0	1.94	3.88	5.64	6.89	5.64	6.2
GS (high acc.)	time (sec.)	655	342	176	122	92	89	78
	speedup	1.0	1.92	3.72	5.37	7.12	7.36	8.40

Table 4. Speedup for parallelized algorithms on the PID tuning problem

		1	2	4	6	8	10	12
1st order	time (sec.)	4995	2575	1343	956	735	651	531
	speedup	1.0	1.94	3.72	5.22	6.80	7.67	9.41
Ncmp (high acc.)	time (sec.)	202	114	58	39	29	24	21
	speedup	1.0	1.77	3.48	5.18	6.97	8.42	9.62
GS (high acc.)	time (sec.)	203	103	53	35	27	22	18
	speedup	1.0	1.97	3.83	5.80	7.52	9.23	11.28

7 Results

It occurred that both version of the proposed modification of the algorithm allow to enclose the Pareto-set far more precisely. The difference for Pareto-fronts was only marginal (but not negligible), but for Pareto-sets (in the decision space) the paving generated by the algorithm using the 2nd order information is more than 3 times smaller (measuring with the Lebesgue measure) for the Kim problem and about 6 times smaller for the PID tuning problem.

The versions using the componentwise Newton operator is computationally intensive for the Kim problem. It seems to be caused by the necessity to compute Hesse matrices (in addition to gradients) using the automatic differentiation arithmetic.

The traditional interval Newton step, based on the Gauss-Seidel operator, requires only one computation of gradients of all functions to prepare to narrow the box. The componentwise operator has to recompute the gradient information in each step of the narrowing operator, i.e., to recompute all gradients each time.

This phenomenon does not affect efficiency for the PID tuning problem, which is probably caused by the following reason. All solutions for this problem lie on the boundaries. The version of the algorithm that uses 1st order derivatives only is not able to delete the interior boxes at the early stage and has to deal with them for several iterations. On the other hand both versions using the second order derivatives are able to delete the interior boxes relatively quickly and most of their work is analyzing the boundaries for which the Newton's method cannot be applied (in our implementation we do not use reduced gradients or Hesse matrices as in [9]) and consequently there is no difference between the different Newton operators used.

Parallelization, as it could be expected, improved the performance of the algorithm. The speedup is satisfactory for 4–6 threads (except the 1st order information only version for the Kim problem), but it usually scales worse further. This is probably caused by relatively time consuming critical sections (seeking for dominated quadruples in a linked list).

8 Conclusions

In the paper we tested two versions of the interval algorithm using the 2nd order information for seeking Pareto-set of a multi-criteria problem. We compared them with interval algorithm based on 1st order information only. It occurred that information enrichment – if processed properly – can improve both efficiency and accuracy of the algorithm several times.

Also, using the Gauss-Seidel interval operator with rectangular matrix seems to be a much better solution than using the componentwise Newton operator; at least for some problems (like the Kim problem).

9 Future Work

There is some more research to be done about the parallelization. It is going to be improved by reducing the critical sections. Instead of the synchronized linear search for quadruples to delete, we are going to store the information about obtained non-dominated points in a separate shared data structure.

Acknowledgments. The research has been supported by the Polish Ministry of Science and Higher Education under grant N N514 416934.

References

1. C-XSC interval library, `http://www.xsc.de`
2. POSIX Threads Programming, `https://computing.llnl.gov/tutorials/pthreads`
3. Barichard, V., Hao, J.K.: Population and Interval Constraint Propagation Algorithm. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 88–101. Springer, Heidelberg (2003)
4. Fernandez, J., Toth, B.: Obtaining an outer approximation of the efficient set of nonlinear biobjective problems. Journal of Global Optimization 38, 315–331 (2007)
5. Haimes, Y.Y.: Risk Modeling, Assessment, and Management. J. Wiley, New York (1998)
6. Hansen, E., Walster, W.: Global Optimization Using Interval Analysis. Marcel Dekker, New York (2004)
7. Herbort, S., Ratz, D.: Improving the efficiency of a nonlinear-system-solver using the componentwise Newton method, `http://citeseer.ist.psu.edu/409594.html`
8. Jaulin, L., Walter, E.: Set Inversion Via Interval Analysis for nonlinear bounded-error estimation. Automatica 29, 1053–1064 (1993)
9. Kearfott, R.B.: Rigorous Global Search: Continuous Problems. Kluwer, Dordrecht (1996)
10. Kearfott, R.B., Nakao, M.T., Neumaier, A., Rump, S.M., Shary, S.P., van Hentenryck, P.: Standardized notation in interval analysis (2002), `http://www.mat.univie.ac.at/~neum/software/int/notation.ps.gz`
11. Kim, I.Y., de Weck, O.L.: Adaptive weighted-sum method for bi-objective optimization: Pareto front generation. Structural and Multidisciplinary Optimization 29, 149–158 (2005)
12. Kubica, B.J.: Interval methods for solving underdetermined nonlinear equations systems. Presented at SCAN, Conference, El Paso, Texas, USA (2008)
13. Kubica, B.J., Woźniak, A.: Interval Methods for Computing the Pareto-Front of a Multicriterial Problem. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Wasniewski, J. (eds.) PPAM 2007. LNCS, vol. 4967, pp. 1382–1391. Springer, Heidelberg (2008)
14. Kubica, B. J., Woźniak, A.: A multi-threaded interval algorithm for the Pareto-front computation in a multi-core environment. Presented at PARA 2008 Conference, accepted for publication in LNCS 6126 (2010)
15. Kubica, B.J., Woźniak, A.: Optimization of the multi-threaded interval algorithm for the Pareto-set computation. Journal of Telecommunications and Information Technology 1, 70–75 (2010)
16. Ruetsch, G.R.: An interval algorithm for multi-objective optimization. Structural and Multidisciplinary Optimization 30, 27–37 (2005)