Success-Guided Selection of Expanded Systems for Result-Verifying Nonlinear Solvers*

Elke Just
Applied Computer Science and Scientific Computing Group, University of Wuppertal, Germany
just@math.uni-wuppertal.de

Bruno Lang
Applied Computer Science and Scientific Computing Group, University of Wuppertal, Germany
lang@math.uni-wuppertal.de

Abstract

Most result-verifying solvers for nonlinear systems include a branch-and-bound algorithmic backbone, complemented with accelerating methods such as constraint propagation, interval Newton, and many others. Often the effectiveness of the accelerators can be improved if one works not only with the given system but also with expanded systems, which are obtained by introducing additional variables for suitable subterms. While reducing the overall number of boxes that must be considered during the solution, applying the accelerators to the (often much larger) expanded systems can increase significantly the time spent on a single box. Therefore a good strategy for deciding which methods to apply to which expanded system is essential for the performance and robustness of the whole solver.

We propose a heuristic that selects expanded systems based on performance information gathered during the computations. Numerical experiments show that this new dynamical strategy tends to be superior to a fixed small-to-large traversal (with restarts) of the expanded systems, which has been the default strategy in our nonlinear solver and optimizer SONIC.

Keywords: interval analysis, expanded systems, solver, nonlinear systems of equations

AMS subject classifications: 65G30, 65G40, 65H10

*Submitted: September 20, 2011; Revised: December 21, 2011; Accepted: March 18, 2012.

1 Introduction

Nonlinear systems arise in a variety of applications, including kinematics, process analysis and design, and automated theorem proving. Often it is desirable or even vital to be sure that all solutions are detected; hence, result-verifying solvers are used [7, 12, 13, 14].

Given a function $f : \mathbb{R}^n \supseteq D \to \mathbb{R}^m$ and a search box $\boldsymbol{x}^s \subseteq D$, a result-verifying solver determines a *covering* of the solution set, i.e., a collection of boxes $\boldsymbol{x}^i \subseteq \boldsymbol{x}^s$, $i \in I$, such that all solutions are retained, $\{x \in \boldsymbol{x} \mid f(x) = 0\} \subseteq \bigcup_i \boldsymbol{x}^i$, and a prescribed precision $\epsilon_j > 0$ is achieved in each coordinate j, $\operatorname{width}(\boldsymbol{x}^i_j) \le \epsilon_j \ \forall i, j$.

Throughout this note, a *box* (or interval vector) $\boldsymbol{x}$ denotes a tuple $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k) \subseteq \mathbb{R}^k$ of appropriate dimension, where the $\boldsymbol{x}_j = [\underline{x}_j, \overline{x}_j]$ are real intervals. The *width* of an interval is $\operatorname{width}(\boldsymbol{x}_j) = \overline{x}_j - \underline{x}_j$, and if $\operatorname{width}(\boldsymbol{x}_j) = 0$ then $\boldsymbol{x}_j$ is also called a *thin* component of $\boldsymbol{x}$.

Interval-based result-verifying nonlinear solvers are usually based on a simple branch-and-bound algorithm, which works as follows. For each component, an *enclosure* $\boldsymbol{f}_i$ for the range of the ith function over the box $\boldsymbol{x}$ is determined, e.g., by plain interval evaluation or more sophisticated methods [1, 11, 15]. If $0 \notin \boldsymbol{f}_i$ for even one enclosure then $\boldsymbol{x}$ cannot contain a solution and therefore is discarded. Otherwise, $\boldsymbol{x}$ is subdivided into two or more subboxes, and these are handled in the same way. The recursion terminates when the current box meets the precision requirement, in which case the box becomes part of the covering. If desired, additional tests may be applied to check whether the box indeed does contain a solution.

In practice, the efficiency of the branch-and-bound algorithm must be improved by adding acceleration techniques such as the interval Newton method [8, 11] or Constraint Propagation [8]. These accelerators often allow contracting the boxes before the recursive calls or even discarding boxes.

Further improvement comes from applying these techniques not only to the given system $f(x) = 0$, but also to *expanded systems*, which are obtained by introducing additional variables for some intermediate quantities, together with their "defining equations." To give an example, we might introduce an additional variable x_{n+1} for the expression $x_1^2 + x_2^2$ and replace each occurence of this expression in the given system with x_{n+1}, while adding the equation $x_{n+1} = x_1^2 + x_2^2$ (or for further computations $x_{n+1} - x_1^2 - x_2^2 = 0$) to the system.

Depending on which expressions are replaced with new variables, different expanded systems can be constructed, and the accelerators may be applied to all or only to some of them. We will call an expanded system *smaller* than another one if it has fewer variables, and *larger* if it has more variables. This does *not* imply that the larger system must contain all the variables of the smaller system, but all systems contain at least the variables of the Original (given) system.

The GlobSol nonlinear solver [9] makes use of the Original and FullSplit systems. The FullSplit system [10, 11] is obtained by replacing *all* intermediate quantities until only elementary constraints with just one unary or binary operator are left; see Fig. 1 for an example.

Since expanded systems can contain significantly more variables and equations than the Original system, the costs for applying the accelerators to them tend to be substantially higher. On the other hand, accelerators such as the Interval Newton method may be much more effective if applied to expanded systems. In [10] Kearfott gives two reasons for this: The expanded system is less nonlinear than the Original system, and the Jacobian is sparse.

ORIGINAL			corresponding FULLSPLIT		
0	$=$	$\sin(x_1^2 + x_2^2)$	x_4	$=$	x_1^2
0	$=$	$(x_1^2 + x_2^2) * x_3^2$	x_5	$=$	x_2^2
0	$=$	$x_1^2 + x_2^2 + x_3^2$	x_6	$=$	$x_4 + x_5$
			x_7	$=$	$\sin(x_6)$
			x_8	$=$	x_3^2
			x_9	$=$	$x_6 * x_8$
			x_{10}	$=$	$x_6 + x_8$
			0	$=$	x_7
			0	$=$	x_9
			0	$=$	x_{10}

Figure 1: Example for a FULLSPLIT expanded system. See also Fig. 2 for a graphical representation.

Besides ORIGINAL and FULLSPLIT, our result-verifying nonlinear solver SONIC (**S**olver and **O**ptimizer for **N**onlinear problems based on **I**nterval **C**omputations) can make use of three additional "intermediate" expanded systems,

COMMONSUBTERMS: Here we add intermediate variables for all "largest" subterms with multiple occurences. This strategy also reduces the number of identical entries in the Jacobian.

COMMONSUBTERMSANDNEIGHBORS: This is a larger version of the COMMONSUBTERMS system. Not only common subterms are replaced with new variables, but also their siblings and parents in the term net.

LINEARIZING: This strategy is motivated by the observation that one advantage of expanded systems is their lower nonlinearity. To achieve this it is sufficient to define intermediate variables just for those subterms that are not atomic or linear.

See Fig. 2 for an example. Even more expanded systems have been considered in [16].

SONIC makes use of constraint propagation, Taylor methods of first and second order, and a hybrid version of the interval Newton method (see [2]) as accelerating devices. Further features of SONIC are a hybrid subdivision strategy, verification methods and a constrained optimizer using the nonlinear solver via the Fritz John conditions. The implementation also provides parallel versions for shared and distributed memory architectures and portability by the possibility to use several basic interval libraries. For more information on SONIC see, e.g., [3] or [2].

As hinted at above, the efficiency of the overall method is determined to a large degree by the choice of which expanded systems to consider at all and which accelerators to apply to each of them. Running just the basic branch-and-bound scheme (no expanded systems, accelerators only on the ORIGINAL system or not at all) maximizes the number of boxes that can be considered per second, but this strategy often leads to prohibitively large overall numbers of boxes. By contrast, running all accelerators on all expanded systems tends to minimize the overall number of boxes to be considered, at the cost of much increased computational effort per box.

Both extreme strategies can make sense. For the Trigonometric test problem from Sect. 4, running all accelerators on all expanded systems leads to an overall time of

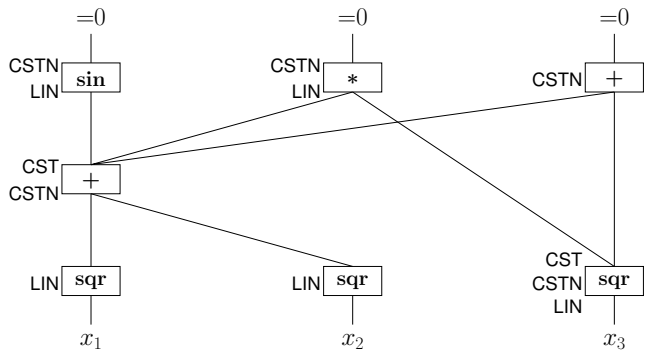

Figure 2: "Term net" for the system from Fig. 1. The labels to the left of each node indicate that the node corresponds to an additional variable in the respective expanded system, CommonSubterms (CST), CommonSubtermsAndNeighbors (CSTN), or Linearizing (LIN). Thus, the CommonSubterms system contains the original variables x_1, x_2, x_3, and two additional variables $x_4 = x_1^2 + x_2^2$, and $x_5 = x_3^2$. In the FullSplit, each node corresponds to a new variable.

149.77 seconds, spent on a total of 12,241 boxes, whereas turning all accelerators off lets the box count and time explode: 46,919,939 boxes and 27,945.10 seconds. Here, the accelerators are essential for success. By contrast, for some of the harder min-t-N systems (representing N-point spherical t-designs) from [4] it was better to go for mere box throughput by disabling the accelerators.

The present note is concerned with strategies for finding an effective compromise between the two extremes. In Sect. 2 we briefly review SONIC's current selection strategy, which relies only on information from the box under consideration. In Sect. 3 a new heuristic is introduced, which also makes use of information gathered during the computations that led to the current box. Numerical experiments presented in Sect. 4 show that this allows us to use the advantages of expanded systems even more effectively.

2 A strategy based on box-internal information

As the cost of the accelerators tends to increase with the size of the system, a promising strategy for applying expensive accelerators selectively is to proceed from small to large expanded systems until "sufficient success" is achieved. Thus, the most expensive methods are tried only if the box cannot be contracted significantly with cheaper methods.

This has been the default strategy in SONIC, complemented with the option to restart the traversal of the hierarchy of expanded systems if the box has been contracted by a user-defined amount; cf. [16, in particular p. 114]. Allowing restarts is motivated by the observation that after a substantial contraction the smaller systems may be helpful again.

The success of contraction is monitored via the "relative volume" of the box,

$$\mathsf{RelVol}(\boldsymbol{x}) := \prod_{\text{width}\,(\boldsymbol{x}_j)\neq 0} \text{width}\,(\boldsymbol{x}_j),$$

and the default threshold for a restart in SONIC is a reduction by a factor of $\kappa = 10^{0.025} \approx 1.06$. Two points should be noted here. First, in practice one computes $\log_{10}(\mathsf{RelVol}(\boldsymbol{x}))$ instead of $\mathsf{RelVol}(\boldsymbol{x})$ to avoid underflow. Second, thin components do not contribute to the relative volume. Reduction of a non-thin component to a thin interval is considered "sufficent success," and therefore the traversal is also restarted whenever the number of thin components, Thin, increases.

Finally, the traversal is aborted if either the box has been contracted to the desired size for a solution box, or the contraction methods have been applied to this box for a prescribed maximum number of times. (In the latter case the box will first be bisected in the overall branch-and-bound algorithm before trying to apply any contraction methods again. This bisecton is done independently from the strategy in use.) The default configuration limits the total number of contraction steps for all expanded systems to 2 times the number of available expanded systems. Thus, the largest system is considered at most twice per box, whereas smaller systems may be contracted more often, depending on when restarts occur.

The resulting algorithm is summarized in Alg. 1, and the sequence of exclusion tests and accelerators applied to each box is given in Alg. 2. Constraint Propagation may be done repeatedly, due to its low cost. Figure 3 depicts the possible branches in Alg. 1 and compares it to a new scheme that will be presented in Sect. 3.

One drawback of Alg. 1 is that it does not make much use of earlier computations. Except for restarts, *all* available expanded systems are considered for each box $\boldsymbol{x}$ in the same small-to-large order. As Willems already mentioned in [16] this may not be the best heuristic. In the following section we extend the basic strategy such that the selection of expanded systems for the current box $\boldsymbol{x}$ is guided by "experience" with boxes that have been processed before reaching $\boldsymbol{x}$.

3 A success-guided heuristic

This heuristic was motivated by a test in which we measured the success in contraction for five test problems and the Original, CommonSubterms, and FullSplit systems. The percentage of computations for which the contraction methods were successful in the father box, but not in the child box, or vice versa, has always been less than 45%, often even less than 20%. This indicates that it may be sensible to choose expanded systems depending on the computations done before in the father box or even ancestor boxes. On the other hand the same experiment showed that for different problems different expanded systems may be most promising, so no uniform strategy can be used for all problems.

To guide the selection, a *"gauge"* quantity γ is associated with each expanded system. The gauge keeps track of the "performance" (product of contraction ratio and computation time) of the accelerators for that particular expanded system.

Note that γ measures the success of contraction and the time needed for the whole expanded system, including Constraint Propagation and Taylor methods. Usually these are cheap, but for some problems they also needed a noticeable part of the computation time, and therefore we control not only calls of the Newton method but also of all methods used for contraction on an expanded system.

Algorithm 1 BOX-BASED STRATEGY FOR TRAVERSING THE HIERARCHY OF EXPANDED SYSTEMS

System := FirstSystem
/* If Newton is enabled for the ORIGINAL system then FirstSystem = ORIGINAL, otherwise it is the next larger expanded system */

repeat
  Determine $\mathsf{Thin}_{\text{old}}$, $\mathsf{RelVol}_{\text{old}}$
  Run exclusion tests and accelerators on current System (Alg. 2)

  /* Which system to consider next? */
  if System $\neq$ FirstSystem **and**
      ($\mathsf{RelVol}_{\text{new}} * \kappa < \mathsf{RelVol}_{\text{old}}$ **or** $\mathsf{Thin}_{\text{new}} > \mathsf{Thin}_{\text{old}}$) **then**
    /* Sufficient contraction for restart */
    System := FirstSystem, $\mathsf{Thin}_{\text{old}} := \mathsf{Thin}_{\text{new}}$, $\mathsf{RelVol}_{\text{old}} := \mathsf{RelVol}_{\text{new}}$
  else if the current box $\boldsymbol{x}$ meets the precision requirements **or**
      System is already the largest system **then**
    /* Box is small enough, or sufficient contraction was impossible */
    Terminate **repeat** loop
  else
    System := next larger expanded system
  end if
until the loop body has been executed a maximum number of times

Algorithm 2 EXCLUSION TESTS AND ACCELERATORS FOR THE CURRENT EXPANDED SYSTEM

Do Constraint Propagation if enabled
if additional order-1 and/or order-2 Taylor contraction is enabled **then**
  Do Taylor contraction(s)
  Do Constraint Propagation if enabled
end if
Do direct evaluation and continuity check
if Interval Newton is enabled **then**
  Do (hybrid) Newton
  Do Constraint Propagation if enabled
end if

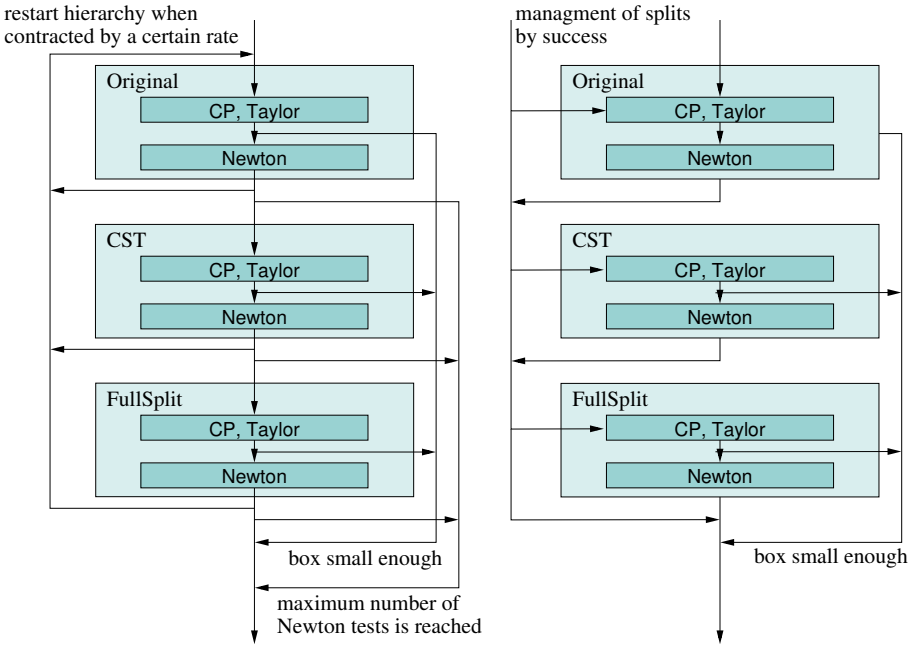

Figure 3: Flow of the box-centered strategy (left) and the success-driven heuristic (right) for using expanded systems. The charts reflect the default setup with three expanded systems (ORIGINAL, COMMONSUBTERMS, and FULLSPLIT).

If the decision whether or not to use a certain expanded system for the *current* box is to be based on these performance indicators then the γ values from *previous* runs must be used, i.e., these must be inherited from the parent box in the branch-and-bound recursion. To obtain smoother behavior we take into account not only the performance in the parent box but also in all boxes along the bisection path from the root to the current box, giving "older" values less weight than more recent ones. This is achieved by averaging the new γ value with the inherited value,

$$\gamma := \beta \cdot \gamma_{\text{new}} + (1 - \beta) \cdot \gamma_{\text{old}} \quad \text{for some fixed } \beta \in [0, 1].$$

(In our experiments $\beta = 0.75$ yielded good results.) If an expanded system is not used in the current box then the value inherited from the parent box is taken, $\gamma := \gamma_{\text{old}}$. To initialize the weighting scheme we let $\gamma_{\text{old}} = 0$ at the root of the branch-and-bound tree (the original search box $\boldsymbol{x}^s$) and disable the selection heuristic within the three topmost recursion levels. That is, in the first three boxes on each path from the bisection root *all* available expanded systems are used.

For the other boxes, the *smallest* available system is used if its earlier performance was not worse than that of the next larger system, whereas an expanded system is used if its earlier performance was at most α times worse than the performance of the smallest system (default: $\alpha = 4$).

Apart from taking previous success into account, our success-driven heuristic differs from Alg. 1 mainly in two aspects. First, our experiments indicated that restarts do not

yield significant additional improvements with the new selection criterion. Therefore we have abandoned them to simplify the algorithmic logic. Second, thin components do not necessarily mark real progress if the user's precision requirements for the respective components are moderate. Therefore we rely on (the logarithm of) a "thresholded volume"

$$\mathsf{ThVol}(\boldsymbol{x}) := \prod_j \mathrm{width}^*(\boldsymbol{x}_j), \quad \text{where } \mathrm{width}^*(\boldsymbol{x}_j) := \begin{cases} \theta_j & \text{if } \overline{x}_j - \underline{x} \le \theta_j \\ \overline{x}_j - \underline{x} & \text{if } \theta_j \le \overline{x}_j - \underline{x} \le \Theta_j \\ \Theta_j & \text{if } \overline{x}_j - \underline{x} \ge \Theta_j \end{cases},$$

taking, e.g., θ_j and Θ_j "sufficiently" small resp. large. With this notion of volume neither thin components nor unbounded components ($\underline{x}_j = -\infty$ and/or $\overline{x}_j = \infty$ are allowed in SONIC) require special treatment, and only one real number per expanded system must be added to the data structure for a box.

The resulting algorithm is summarized in Alg. 3; see also Fig. 3 for a graphical comparison to Alg. 1.

Algorithm 3 SUCCESS-DRIVEN HEURISTIC FOR USING EXPANDED SYSTEMS FOR A GIVEN BOX

```
1: /* FirstSystem is determined as in Alg. 1, and the available expanded systems
      are numbered consecutively from small to large */
2: for System := FirstSystem to largest system in small-to-large order do
3:    if (recursion level < 4) or
            (System = FirstSystem and γ[FirstSystem] ≤ γ[FirstSystem + 1]) or
            (System ≠ FirstSystem and γ[System] < α · γ[FirstSystem]) then
4:       Determine ThVol_old and T_start := current time
5:       Run exclusion tests and accelerators on current System (Alg. 2)
6:       Determine ThVol_new and T_end := current time
7:       γ_new := (T_end − T_start) · (ThVol_new/ThVol_old)
8:       γ[System] := β · γ_new + (1 − β) · γ[System] for some fixed β ∈ [0, 1]
9:    end if
10: end for
```

The success-driven heuristic implies a somewhat non-deterministic behavior. Since the time spent for each expanded system depends – besides other factors – on the actual work load of the machine, different runs will lead to different gauge values, and therefore different expanded systems may be chosen. Thus the box counts and times, and even the size and number of solution boxes may be different in each run with this strategy. In all of our tests, however, times and box numbers differed just slightly (usually less than 1 per cent, never more than 5 per cent). If a deterministic behavior is desired, e.g. for debugging purposes, one can still resort to the old strategy via SONIC's runtime configuration file.

4 Numerical experiments

Numerical experiments with a test set consisting of 14 problems of widely varying difficulty were made to compare SONIC's selection strategy (with default parameter settings) to our new heuristic; see Table 1. For both strategies we used the default

hierarchy of SONIC, that is, ORIGINAL, COMMONSUBTERMS, and FULLSPLIT. As this note focuses on the "exclusion and contraction" phase, the subsequent checks for verifying the existence of zeros were turned off. The timings were obtained by a serial computation on an Intel(R) Core(TM)2 Quad CPU Q9650 @ 3.00GHz with C-XSC 2.2.4.

Problem	boxes considered		overall time in s	
	SONIC	new scheme	SONIC	new scheme
min-04-06 [4]	785	809	12.17	8.64
Griewank [2]	2175	2175	12.60	6.55
Reactor [2]	549	577	27.92	12.40
Agrawal_Hopf_1	5889	6907	48.85	37.51
Brent, $n = 7$ [6]	51851	55421	79.09	29.28
Eco9 [2], [6]	30109	30929	85.78	55.19
Trigexp1 [6]	1385	1405	105.57	31.73
Trigonometric, $n = 10$ [6]	18569	31751	132.70	133.87
Agrawal_SaddleNode_2	49195	49201	185.85	88.16
DirectKinematics [2]	9157	10401	189.83	53.32
DesignProblem9 [2], [6]	27949	44497	195.02	112.15
7erSystem [2]	11523	17375	446.28	79.67
min-04-07 [4]	68185	68699	1894.40	1408.86
Agrawal_Hopf_2	2959459	2976919	32902.30	13886.70

Table 1: Comparison of SONIC's default scheme and the new heuristic for using three expanded systems. (The Agrawal systems come from modelling different singularities in process design, cf. [13].)

We see that we achieved a major decrease in running time for some problems and that there is no problem for which the computation time increases much. By contrast, box numbers usually increase. This was to be expected because we do not always use the most powerful methods and therefore may have to consider more boxes, but with potentially cheaper methods. For some problems such as DesignProblem9 we see a significantly higher box number, but still we get faster execution with our new heuristic.

Over the whole test set, the arithmetic mean of the ratio of new and old values is 1.17 for the box numbers and 0.53 for the time. Thus, on average we need to consider roughly 20 per cent more boxes, but we can save nearly 50 per cent computation time.

The experiments therefore gave evidence that our new heuristic is as good as or superior to the original traversal of the expanded systems using only box-internal information.

We also ran SONIC with all five expanded systems enabled, together with the new heuristic. For most of the problems from the test set, the results differed little from those in Table 1, with a slight superiority of the 3-systems variant. However, this does not imply that the COMMONSUBTERMSANDNEIGHBORS and LINEARIZING systems were never called, or not successful. More detailed timings revealed that for the Trigonometric problem these two expanded systems led to dismissal or substantial contraction of the boxes in most cases they were used, but the overall time spent in

System	Variables	Strategy	Calls	Dismissals	Contractions	Time
ORIGINAL	10	new(3)	20670	3576	4171	20.22
		new(5)	18447	824	2980	20.03
CST	23	new(3)	13216	4657	8276	45.67
		new(5)	12553	4573	7903	39.19
LINEARIZING	30	new(3)	–	–	–	–
		new(5)	580	25	386	1.67
CSTN	81	new(3)	–	–	–	–
		new(5)	1214	155	1058	6.56
FULLSPLIT	86	new(3)	7693	32	5599	17.49
		new(5)	2719	269	1923	6.76

Table 2: Comparison of the new heuristic, applied to three or five expanded systems (ORIGINAL, COMMONSUBTERMS, FULLSPLIT, plus LINEARIZING and COMMONSUBTERMSANDNEIGHBORS for the 5-systems runs) of the Trigonometric problem.

the accelerators remained almost identical to that of the 3-systems run; see Table 2.

While SONIC's 3-system default hierarchy performed consistently close to optimum in these tests, there are other problems for which the new heuristic with a combination of ORIGINAL, FULLSPLIT, and one of the other intermediate expanded systems was even better – but so far we are able to detect such behavior only a posteriori.

Systematic comparisons with other solvers are out of the scope of this paper. Tests with GlobSol ([9], using the version of 2003 and a constant objective function) and the benchmarks of ALIAS [5] suggest that SONIC with the success-driven strategy is competitive with respect to overall timings and box numbers.

5 Conclusions and perspectives

We have compared two different methods for selecting which expanded systems should be used to apply contraction methods.

In our new heuristic this decision is based on former success in contracting the ancestors of the current box. Numerical experiments gave evidence that this strategy is superior to just relying on box-internal information.

As already mentioned our investigations have concentrated mainly on a setup with three expanded systems: ORIGINAL, COMMONSUBTERMS, and FULLSPLIT. Besides further improvements of the success-driven heuristic, future work might address the question of which expanded systems should be considered at all for a given problem: Should we use more (or others) than those three, should the available expanded systems be kept constant or be allowed to change during the computations, and which features of the given system or the behavior of the algorithm can be used to make these decisions in an automated way?

Acknowledgements

The authors want to thank the referees for their valuable comments, which helped to improve the presentation.

References

[1] Götz Alefeld and Jürgen Herzberger. *Introduction to Interval Computations*. Academic Press, New York, 1983.

[2] Thomas Beelitz. *Effiziente Methoden zum verifizierten Lösen von Optimierungsaufgaben und nichtlinearen Gleichungssystemen*. PhD thesis, Bergische Universität Wuppertal, Germany, 2006.

[3] Thomas Beelitz, Christian H. Bischof, Bruno Lang, and Paul Willems. SONIC—A framework for the rigorous solution of nonlinear problems. Report BUW-SC 04/7, Fachbereich Mathematik und Naturwissenschaften, Bergische Universität Wuppertal, 2004.

[4] Thomas Beelitz, Bruno Lang, Peer Ueberholz, and Paul Willems. Closing the case $t = 3$ for 3D spherical t-designs using a result-verifying nonlinear solver. *Reliab. Comput.*, 14:66–77, June 2010.

[5] ALIAS Benchmarks. http://www-sop.inria.fr/coprin/logiciels/ALIAS/Benches/benches.html.

[6] COPRIN Homepage. http://www sop.inria.fr/coprin/index_english.html.

[7] Andreas Frommer. Proving conjectures by use of interval arithmetic. In Ulrich Kulisch, Rudolf Lohner, and Axel Facius, editors, *Perspectives on Enclosure Methods*, pages 1–13. Springer, Wien, 2001.

[8] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis, with Examples in Parameter and State Estimation, Robust Control and Robotics*. Springer-Verlag, London, 2001.

[9] R. B. Kearfott. GlobSol user guide. *Optimization Methods and Software 24 (4-5)*, pages 687–708, August, 2009.

[10] R. Baker Kearfott. Decomposition of arithmetic expressions to improve the behavior of interval iteration for nonlinear systems. *Computing*, 47:169–191, 1991.

[11] R. Baker Kearfott. *Rigorous Global Search: Continuous Problems*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1996.

[12] Jean-Pierre Merlet. Interval analysis for certified numerical solution of problems in robotics. *Int. J. Appl. Math. Comput. Sci.*, 19(3):399–412, 2009.

[13] Martin Mönnigmann, Wolfgang Marquardt, Christian H. Bischof, Thomas Beelitz, Bruno Lang, and Paul Willems. A hybrid approach for efficient robust design of dynamic systems. *SIAM Rev.*, 49(2):236–254, 2007.

[14] Yusuke Nakaya and Shin'ichi Oishi. Finding all solutions of nonlinear systems of equations using linear programming with guaranteed accuracy. *Journal of Universal Computer Science*, 4(2):171–177, 1998.

[15] Arnold Neumaier. *Interval Methods for Systems of Equations*. Cambridge University Press, Cambridge, UK, 2000.

[16] Paul Willems. Symbolisch-numerische Techniken zur verifizierten Lösung nichtlinearer Gleichungssysteme, 2004.