Efficient pruning technique based on linear relaxations

(in Proc. of COCOS'03 LNCS xxx, 2005)

Yahia Lebbah[2,1], Claude Michel[1], and Michel Rueher[1]

[1] {cpjm, rueher}@essi.fr
COPRIN (I3S/CNRS - INRIA),
Université de Nice–Sophia Antipolis,
930, route des Colles, B.P. 145, 06903 Sophia Antipolis Cedex, France
[2] ylebbah@sophia.inria.fr
Université d'Oran Es-Senia,
Faculté des Sciences, Département Informatique,
B.P. 1524 El-M'Naouar, Oran, Algeria

Abstract. This paper extends the Quad-filtering algorithm for handling general nonlinear systems. This extended algorithm is based on the RLT (Reformulation-Linearization Technique) schema. In the reformulation phase, tight convex and concave approximations of nonlinear terms are generated, that's to say for bilinear terms, product of variables, power and univariate terms. New variables are introduced to linearize the initial constraint system. A linear programming solver is called to prune the domains. A combination of this filtering technique with Box-consistency filtering algorithm has been investigated. Experimental results on difficult problems show that a solver based on this combination outperforms classical CSP solvers.

1 Introduction

Numerical constraint systems are widely used to model problems in numerous application areas ranging from robotics to chemistry. Solvers of nonlinear constraint systems over the real numbers are based upon partial consistencies and searching techniques.

The drawback of classical local consistencies (e.g. 2B-consistency [13] and Box-consistency [3]) comes from the fact that the constraints are handled independently and in a blind way. 3B-consistency [13] and kB-consistency [13] are partial consistencies that can achieve a better pruning since they are "less local" [10]. However, they require numerous splitting steps to find the solutions of a system of nonlinear constraints; so, they may become rather slow.

For instance, classical local consistencies do not exploit the semantic of quadratic terms; that's to say, these approaches do not take advantage of the very specific semantic of quadratic constraints to reduce the domains of the variables. Linear programming techniques [1, 25, 2] do capture most of the semantic

of quadratic terms (e.g., convex and concave envelopes of these particular terms). That's why we have introduced in [11] a global filtering algorithm (named `Quad`) for handling systems of quadratic equations and inequalities over the real numbers. The `Quad`-algorithm computes convex and concave envelopes of bilinear terms xy as well as concave envelopes and convex underestimations for square terms x^2.

In this paper, we extend the `Quad`-framework for tackling general nonlinear system. More precisely, since every nonlinear term can be rewritten as sums of products of univariate terms, we introduce relaxations for handling the following terms:

- power term x^n
- product of variables $x_1 x_2 ... x_n$
- univariate term $f(x)$

The `Quad`-algorithm is used as a global filtering algorithm in a branch and prune approach [29]. Branch and prune is a search-tree algorithm where filtering techniques are applied at each node. `Quad`-algorithm uses Box-consistency and 2B-consistency filtering algorithms. In addition, linear and nonlinear relaxations of non-convex constraints are used for range reduction in the branch-and-reduce algorithm [19]. More precisely, the `Quad`-algorithm works on the relaxations of the nonlinear terms of the constraint system whereas `Box`-consistency algorithm works on the initial constraint system.

Yamamura et. al. [31] have first used the simplex algorithm on quasi-linear equations for excluding interval vectors (boxes) containing no solution. They replace each nonlinear term by a new variable but they do not take into account the semantic of nonlinear terms[3]. Thus, their approach is rather inefficient for systems with many nonlinear terms.

The paper is organised as follows. Notations and classical consistencies are introduced in section 2. Section 3 introduces and extends the `Quad` pruning algorithm. Experimental results are reported in section 4 whereas related works are discussed in section 5.

2 Notation and basics on classical continuous consistencies

This paper focuses on CSPs where the domains are intervals and the constraints are continuous. A n-ary continuous constraint $C_j(x_1, \ldots, x_n)$ is a relation over the reals. $\mathcal{C}$ stands for the set of constraints.

D_x denotes the domain of variable x, that's to say, the interval $[\underline{x}, \overline{x}]$ of allowed values for x. $\mathcal{D}$ stands for the set of domains of all the variables of the considered constraint system.

We use the "reformulation-linearization technique" notations introduced in [25, 2] with some modifications. Let E be some nonlinear expression, $[E]_L$ denotes the set of linear terms coming from a linearization process of E.

[3] They introduce only some weak approximation for convex and monotone functions.

We also use two local consistencies derived from Arc-consistency [14]: 2B-consistency and Box-consistency.

2B-consistency [13] states a local property on the bounds of the domains of a variable at a single constraint level. Roughly speaking, a constraint c is 2B-consistent if, for any variable x, there exists values in the domains of all other variables which satisfy c when x is fixed to $\underline{x}$ or $\overline{x}$.

Box-consistency [3] is a coarser relaxation of Arc-consistency than 2B-consistency. It mainly consists of replacing every existentially quantified variables but one with its interval in the definition of 2B-consistency. Box-consistency [3] is the most successful adaptation of arc-consistency [14] to constraints over the real numbers. Furthermore, the narrowing operator for the Box-consistency has been extended [29] to prove the unicity of a solution in some cases.

The success of 2B-consistency and Box-consistency depends on the precision of enforcing local consistency of each constraint on each variable lower and upper bounds. Thus they are very local and do not exploit any specific semantic of the constraints.

3B-consistency and kB-consistency are partial consistencies that can achieve a better pruning since they are "less local" [10]. However, they require numerous splitting steps to find the solutions of a system of nonlinear constraints; so, they may become rather slow.

3 Using linear relaxations to prune the domains

In this section, we introduce the filtering procedure we propose for handling general constraints. The Quad filtering algorithm (see Algorithm 1.1) consists of three main steps: reformulation, linearization and pruning.

The reformulation step generates $[\mathcal{C}]_R$, the set of implied linear constraints. More precisely, $[\mathcal{C}]_R$ contains linear inequalities that approximate the semantic of nonlinear terms of $[\mathcal{C}]$.

The linearization process first decomposes each non linear term E in sums and products of univariate terms. Then, it replaces nonlinear terms with their associated new variables. For example, consider $E = \{x_2x_3x_4^2(x_6+x_7)+\sin(x_1)(x_2x_6-x_3) = 0\}$, a simple linearization transformation may yield the following sets:

- $[E]_L = \{y_1 + y_3 = 0, y_2 = x_6 + x_7, y_4 = y_5 - x_3\}$
- $[E]_{LI} = \{y_1 = x_2x_3x_4^2y_2, y_3 = \sin(x_1)y_4, y_5 = x_2x_6\}$.

$[E]_{LI}$ denotes the set of equalities that keep the link between the new variables and the nonlinear terms.

Finally, the linearization step computes the set of final linear inequalities and equalities $LR = [\mathcal{C}]_L \cup [\mathcal{C}]_R$, the linear relaxation of the original constraints $\mathcal{C}$.

The pruning step is just a fixed point algorithm that calls a linear programming solver iteratively to reduce the upper and the lower bound of each initial variable. The algorithm terminates when the maximum achieved reduction is smaller than a non-null predetermined threshold ϵ.

Function `Quad_filtering`(`IN`: $\mathcal{X}$, $\mathcal{D}$, $\mathcal{C}$, ϵ) **return** $\mathcal{D}'$
% $\mathcal{X}$: initial variables ; $\mathcal{D}$: input domains; $\mathcal{C}$: constraints; ϵ: minimal reduction, $\mathcal{D}'$: output domains

1. *Reformulation*: generation of linear inequalities $[\mathcal{C}]_R$ for the nonlinear terms in $\mathcal{C}$.
2. *Linearization*: linearization of the whole system $[\mathcal{C}]_L$.
   We obtain a linear system $LR = [\mathcal{C}]_L \cup [\mathcal{C}]_R$.
3. $\mathcal{D}' := \mathcal{D}$
4. *Pruning* :
   While the reduction amount of some bound is greater than ϵ **and** $\emptyset \notin \mathcal{D}'$ **Do**
   (a) Update the coefficients of the linearizations $[\mathcal{C}]_R$ according to the domain $\mathcal{D}'$
   (b) Reduce the lower and upper bounds $\underline{D'_i}$ and $\overline{D'_i}$ of each *initial* variable $x_i \in \mathcal{X}$ by computing min and max of x_i subject to LR with a linear programming solver.

Algorithm 1.1. The `Quad` algorithm

Now, we are in position to introduce the reformulation of nonlinear terms. Section 3.1 recalls the relaxations for the simplest case of bilinear term xy, the product of two distinct variables. Relaxations for the power term are given in section 3.2. The process for approximating general product terms is given in section 3.3. Finally, in section 3.4, we introduce a procedure to relax some univariate terms.

3.1 Bilinear terms

In the case of bilinear terms xy, Al-Khayal and Falk [1] showed that convex and concave envelopes of xy over the box $[\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}]$ can be approximated by the following relations:

$$[xy]_R \;=\; \begin{cases} BIL1 \equiv [(x-\underline{x})(y-\underline{y}) \geq 0]_L \\ BIL2 \equiv [(x-\underline{x})(\overline{y}-y) \geq 0]_L \\ BIL3 \equiv [(\overline{x}-x)(y-\underline{y}) \geq 0]_L \\ BIL4 \equiv [(\overline{x}-x)(\overline{y}-y) \geq 0]_L \end{cases} \tag{1}$$

BIL1 and BIL3 define a convex envelope of xy whereas BIL2 and BIL4 define a concave envelope of xy over the box $[\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}]$. Thus, these relaxations are the optimal convex/concave outer-estimations of xy.

3.2 Power terms

First let us consider square terms. The term x^2 with $\underline{x} \leq x \leq \overline{x}$ is approximated by the following relations:

$$L1(\alpha) \equiv [(x-\alpha)^2 \geq 0]_L \text{ where } \alpha \in [\underline{x}, \overline{x}] \quad (2)$$

$$L2 \equiv [(\underline{x}+\overline{x})x - y - \underline{x}\overline{x} \geq 0]_L \quad (3)$$

Note that $[(x-\alpha)^2 = 0]_L$ generates the tangent line to the curve $y = x^2$ at the point $x = \alpha$. Actually, `Quad` computes only $L1(\overline{x})$ and $L1(\underline{x})$. Consider for instance the quadratic term x^2 with $x \in [-4, 5]$. Figure 1 displays the initial curve (i.e., D_1), and the lines corresponding to the equations generated by the relaxations: D_2 for $L1(-4) \equiv y + 8x + 16 \geq 0$, D_3 for $L1(5) \equiv y - 10x + 25 \geq 0$, and D_4 for $L2 \equiv -y + x + 20 \geq 0$.

We may note that $L1(\overline{x})$ and $L1(\underline{x})$ are underestimations of x^2 whereas $L2$ is an overestimation. $L2$ is also the concave envelope, which means that it is the optimal concave overestimation.

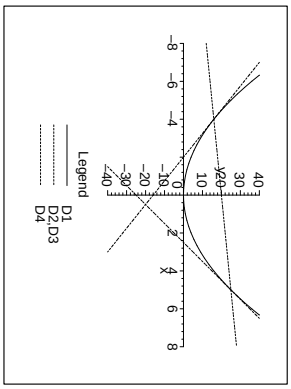

Fig. 1. Approximation of x^2

More generally, a power term of the form x^n can be approximated by $n+1$ inequalities with a procedure proposed by Sherali and Tuncbilek [27], called "bound-factor product RLT constraints". It is defined by the following formula:

$$[x^n]_R \;=\; \{[(x-\underline{x})^i(\overline{x}-x)^{n-i} \geq 0]_L, i = 0...n\} \quad (4)$$

The essential observation is that this relaxation generates tight relations between variables on their upper and lower bounds. More precisely, suppose that some original variable takes a value equal to either of its bounds. Then, all the corresponding new RLT linearization variables that involve this original variable take a relative value that conform with actually fixing this original variable at each of its particular bound in the nonlinear expressions represented by these

new RLT variables [27].
Note that relaxations (4) of the power term x^n are expressed with x^i for all $i \leq n$, and thus provide a fruitful relationship on problems containing many power terms involving the same variable.

The univariate term x^n is convex when n is even, or when n is odd and the value of x is negative; it is concave when n is odd and the value of x is positive. Section 3.4 details the process for handling such convex and concave univariate term. Sahinidis and Twarmalani [21] have introduced the convex and concave envelopes when n is odd by taking the point where the power term x^n and its under-estimator have the same slope. These convex/concave relaxations on x^n are expressed with only $[x^n]_L$ and x. In other words, they do not generate any relations with x^i for $1 < i < n$. That's why we suggest to implement these formulas (4).

Note that for the case $n = 2$, (4) provides the concave envelope.

3.3 Product terms

For the product term

$$x_1 x_2 ... x_n \tag{5}$$

we use a two steps procedure: quadrification and bilinear relaxations.

The *Quadrification* step brings back the multi-linear term into a set of quadratic terms as follows

$$\begin{array}{rcl}
\underbrace{x_1x_2...x_n} & = & \underbrace{x_1...x_{d1}} \quad \underbrace{x_{d1+1}...x_n} \\
x_{1...n} & = & x_{1...d1} \times x_{d1+1...n} \\
\hline
 & & \underbrace{x_1...x_{d2}} \quad \underbrace{x_{d2+1}...x_{d1}} \\
x_{1...d1} & = & x_{1...d2} \times x_{d2+1...d1} \\
\hline
 & & \underbrace{x_{d1+1}...x_{d3}} \quad \underbrace{x_{d3+1}...x_n} \\
x_{d1+1...n} & = & x_{d1+1...d3} \times x_{d3+1...n} \\
\hline
... & & \\
\hline
\end{array}$$

where $x_{i...j} = [x_i x_{i+1} ... x_j]_L$.
For instance, consider the term $x_1x_2x_3x_4x_5$. The proposed quadrification process would operate in the following way:

$$\begin{array}{rcl}
\underbrace{x_1x_2x_3x_4x_5} & = & \underbrace{x_1x_2x_3} \quad \underbrace{x_4x_5} \\
y_1 & = & y_2 \times y_3 \\
\hline
 & & \underbrace{x_1x_2} \quad \underbrace{x_3} \\
y_2 & = & y_4 \times x_3 \\
\hline
 & & \underbrace{x_4} \quad \underbrace{x_5} \\
y_3 & = & x_4 \times x_5 \\
\hline
 & & \underbrace{x_1} \quad \underbrace{x_2} \\
y_4 & = & x_1 \times x_2 \\
\hline
\end{array}$$

So, this quadrification is performed by recursively decomposing each product $x_i...x_j$ into two products $x_i...x_d$ and $x_{d+1}...x_j$. Of course, there are many ways to choose the position of d. Sahnidis et al. [20, 22] use what they call `rAI`, "recursive interval arithmetic", which is a recursive quadrification where $d = j - 1$. We use the middle heuristic `Qmid`, where $d = (i+j)/2$, to obtain balanced degrees on the generated terms. Note that $[E]_{RI}$ contains the set of equalities that transforms a product term E into a set of quadratic identities.

The second step consists in a *Bilinear relaxation* $[[\mathcal{C}]_{RI}]_R$ of all the quadratic identities in $[\mathcal{C}]_{RI}$ with the bilinear relaxations introduced in sub-section 3.1.

Sherali and Tuncbilek [27] have proposed a direct reformulation/linearization technique (RLT) of the whole polynomial constraints without quadrifying the constraints. Applying RLT on the product term $x_1x_2...x_n$ generates the following n-ary inequalities [4] :

$$\prod_{i\in J_1}(x_i - \underline{x_i}) \prod_{i\in J_2}(\overline{x_i} - x_i) \geq 0, \forall J_1, J_2 \subseteq \{1,\ldots,n\} \colon |J_1 \cup J_2| = n \qquad (6)$$

where {1, ..., n} is to be understood as a multi-set and where J_1 and J_2 are multi-sets.

Proposition 1 bounds the number of new variables and relaxations respectively generated by the quadrification and RLT process on the product term (5).

Proposition 1.
Let $T \equiv x_1x_2\ldots x_n$ be some product of degree $n \geq 1$ with n distinct variables. The RLT of T will generate up to $(2^n - n - 1)$ new variables and 2^n inequalities whereas the quadrification of T will only generate $(n-1)$ new variables and $4(n-1)$ inequalities.

Proof: The number of terms of length i is clearly the number of combinations of i elements within n elements, that's to say C_n^i. In the RLT relaxations (6), we generate new variables for all these combinations. Thus, the number of variables is bounded by $\sum_{i=2\ldots n} C_n^i = \sum_{i=0\ldots n} C_n^i - n - 1$, that's to say $2^n - n - 1$ since $\sum_{i=0\ldots n} C_n^i = 2^n$. In (6), Dietmaier considers for each variable alternatively lower and upper bound, thus there are 2^n new inequalities.

For the quadrification process, the proof can be done by induction. For $n = 1$, the formula is true. Now, suppose that for length i (with $1 \leq i < n$), $(i-1)$ new variables are generated. For $i = n$, we can split the term at the position d with $1 \leq d < n$. It results from the induction hypothesis that we have $d-1$ new variables for the first part, and $n-d-1$ new variables for the second part, plus one more new variable for the whole term. So, $n-1$ new variables are generated. Bilinear terms require four relaxations, thus we get $4(n-1)$ new inequalities.

[4] Linearizations proposed in RLT on the whole polynomial problem are built on every non-ordered combination of δ variables, where δ is the highest polynomial degree of the constraint system.

□

Sherali and Tuncbilek [26] have proven that RLT yields a tighter linearization than quadrification on general polynomial problems. However, since the number of generated linearizations with RLT grows in an exponential way, this approach may become very expensive in time and space for non trivial polynomial constraint systems.

Proposition 2 states that quadrification with bilinear relaxations provides convex and concave envelopes with any d. This property results from the proof given in [20] for the `rAI` heuristic.

Proposition 2.
Let $x_1x_2 \dots x_n$ be some product of degree $n \geq 2$ with n distinct positive variables $x_i \in \mathbb{R}_+, i = 1...n$. Then $[[x_1x_2...x_n]_{RI}]_R$ provides convex and concave envelopes of the product term $x_1x_2...x_n$.

Generalisation for sums of products –the so-called multi-linear terms – have been studied recently [4, 23, 17, 20]. It is well known that finding the convex or concave envelope of a multi-linear term is a NP hard problem [4]. The most common method of linear relaxation of multi-linear terms is based on the simple product term. However, it is also well known that this approach leads to a poor approximation of the linear bounding of the multi-linear terms. Sherali [23] has introduced formulae for computing convex envelopes of the multi-linear terms. It is based on an enumeration of vertices of a pre-specified polyhedra which is of exponential nature. Rikun [17] has given necessary and sufficient conditions for the polyhedrality of convex envelopes. He has also provided formulae of some faces of the convex envelope of a multi-linear function. To summarize, it is difficult to characterize convex and concave envelopes for general multi-linear terms. Conversely, the approximation of "product of variables" is an effective approach; moreover, it is easy to implement [22, 21].

3.4 Univariate terms

Here, we provide some relaxations to handle some univariate terms. An overestimation of a convex univariate function f is given by the following envelope:

$$[f(x)]_R \;=\; [f(\underline{x}) + \frac{f(\overline{x}) - f(\underline{x})}{\overline{x} - \underline{x}}(x - \underline{x}) \geq f(x)]_L \tag{7}$$

To underestimate a convex function, we could use the `sandwich` algorithm recently analyzed by Rote [18] and which has been extended by Sahinidis and Twarmalani [22, 21]. Outer estimation of concave functions is based on the following observation : if f is a concave function, then $-f$ is a convex function.

To relax general non-convex functions, splitting is required to identify the convex and concave regions where the above relaxation can be used. To avoid branching, different techniques have been proposed. In the RLT framework [24, 28] many polynomial relaxations have been proposed for bounding univariate terms. These polynomial relaxations are then linearized with RLT techniques.

4 Experimental results

This section reports experimental results on twenty standard benchmarks on which the extended version of `Quad` has been evaluated. Benchmarks `eco6`, `katsura5`, `katsura6`, `katsura7`, `tangents2`, `ipp`, `assur44`, `cyclic5`, `tangents0`, `chemequ`, `noon5`, `geneig`, `kinema`, `reimer5`, `camera1s` were taken from Verschelde's web site [30], `kin2` from [29], `didrit` from [5] (page 125), `lee` from [12], and finally `yama194, yama195, yama196` from [31]. The most challenging benchmark is `stewgou40` [6]. It describes a Gough-Stewart platform with variations on the initial position of the robot as well as on its geometry. The constraint system consists of 9 equations with 9 variables. They express the length of the rods as well as the distances between the connection points.

			BP(Box+Quad(Qmid))			BP(Box)			*Realpaver*	
Name	n	δ	*nSols*	*nSplits*	$T(s)$	*nSols*	*nSplits*	$T(s)$	*nSols*	$T(s)$
cyclic5	5	5	10(10)	650	69.61	10(10)	13373	26.33	10	291.64
eco6	6	3	4(4)	1069	15.69	4(4)	1736	3.73	4	1.26
tangents2	6	2	24(24)	197	39.06	24(24)	14104	27.92	24	16.48
assur44	8	3	10(10)	74	68.11	10(10)	15848	72.55	10	72.56
geneig	6	3	10(10)	5053	417.86	10(10)	290711	868.64	10	475.65
ipp	8	2	10(10)	34	6.82	10(10)	4649	13.96	10	16.80
katsura5	6	2	15(11)	56	10.74	41(11)	8181	12.66	12	6.69
katsura6	7	2	44(28)	503	142.85	182(24)	136597	281.43	32	191.76
kin2	8	2	10(10)	40	7.40	10(10)	3463	19.27	10	2.61
noon5	5	3	11(11)	107	19.65	11(11)	50165	58.69	11	39.01
camera1s	6	2	16(16)	8318	452.97	2(2)	3027924	–	0	–
didrit	9	2	4(4)	90	17.39	4(4)	51284	132.94	4	94.60
kinema	9	2	8(8)	221	25.36	15(7)	244040	572.42	8	268.40
katsura7	8	2	49(43)	1729	831.96	180(35)	1421408	–	44	4675.59
lee	9	2	4(4)	491	54.56	0(0)	2091946	–	0	–
reimer5	5	6	24(24)	132	79.53	24(24)	2230187	2982.92	24	734.10
stewgou40	9	4	40(40)	1538	874.64	6(6)	779925	–	4	–
yama195	60	3	3(3)	6	114.84	0(0)	4997	–	0	–
yama196	30	1	16(0)	108	31.44	0(0)	206900	–	0	–

Table 1. Experimental results: comparing `Quad` and Constraint solvers

The experimental results are reported in Tables 1 and 2. Column n (resp. δ) shows the number of variables (resp. the maximum polynomial degree). Experimentations with `BP(X)`, which stands for a *Branch and Prune* solver based on the `X` filtering algorithm, have been performed with the implementation of `iCOs` [5]. `Quad(H)` denotes the `Quad` algorithm where bilinear terms are relaxed

[5] See http://www-sop.inria.fr/coprin/ylebbah/icos

			BP(Box+Simplex)			BP(Box+Quad(Qmid))			BP(Box+Quad(rAI))		
Name	n	δ	*nSols*	*nSplits*	$T(s)$	*nSols*	*nSplits*	$T(s)$	*nSols*	*nSplits*	$T(s)$
cyclic5	5	5	10(10)	15830	99.98	10(10)	650	69.61	10(10)	660	96.78
eco6	6	3	4(4)	1073	6.44	4(4)	1069	15.69	4(4)	1069	15.74
tangents2	6	2	24(24)	13833	170.58	24(24)	197	39.06	24(24)	197	38.75
assur44	8	3	10(10)	15550	669.83	10(10)	74	68.11	10(10)	74	68.00
geneig	6	3	10(10)	258385	3862.20	10(10)	5053	417.86	10(10)	5053	420.04
ipp	8	2	10(10)	3151	71.24	10(10)	34	6.82	10(10)	34	6.86
katsura5	6	2	41(11)	7731	87.17	15(11)	56	10.74	15(11)	56	10.70
katsura6	7	2	182(24)	134468	2071.93	44(28)	503	142.85	44(28)	503	142.47
kin2	8	2	10(10)	2849	75.20	10(10)	40	7.40	10(10)	40	7.42
noon5	5	3	11(11)	49606	427.28	11(11)	107	19.65	11(11)	107	19.51
camera1s	6	2	2(2)	607875	–	16(16)	8318	452.97	16(16)	8318	451.43
didrit	9	2	4(4)	5361	149.03	4(4)	90	17.39	4(4)	90	17.38
kinema	9	2	14(6)	93248	1885.50	8(8)	221	25.36	8(8)	221	24.98
katsura7	8	2	37(3)	353735	–	49(43)	1729	831.96	49(43)	1729	830.86
lee	9	2	4(4)	129374	3695.48	4(4)	491	54.56	4(4)	491	54.45
reimer5	5	6	2(2)	959267	–	24(24)	132	79.53	24(24)	132	79.79
stewgou40	9	4	6(6)	115596	–	40(40)	1538	874.64	40(40)	1553	990.00
yama195	60	3	3(3)	12	41.69	3(3)	6	114.84	3(3)	6	113.92
yama196	30	1	16(0)	108	31.40	16(0)	108	31.44	16(0)	108	31.45

Table 2. Experimental results: comparing solvers based on different relaxations

with formulas (1), power terms with formulas (4) and product terms with the quadrification method; `H` stands for the heuristic used for decomposing terms in the quadrification process.

The relaxations of univariate functions that have been introduced in section 3.4 have not been exploited, except for the one of the power terms through (4).

The performances of the following five solvers have been investigated:

1. `RealPaver` : a free[6] *Branch and Prune* solver that dynamically combines optimised implementations of `Box`-consistency filtering and `2B`-consistency filtering algorithms [8]
2. `BP(Box)`: a *Branch and Prune* solver based on `Box`-consistency, the ILOG[7] commercial implementation of `Box`-consistency
3. `BP(Box+simplex)`: a *Branch and Prune* solver based on `Box`-consistency and a simple linearization of the whole system without introducing outer-estimations of the nonlinear terms
4. `BP(Box+Quad(Qmid))`: a *Branch and Prune* solver which combines `Box`-consistency algorithm and the `Quad` algorithm where product terms are relaxed with the `Qmid` heuristic

[6] See http://www.sciences.univ-nantes.fr/info/perso/permanents/granvil/realpaver/-main.html

[7] See http://www.ilog.com/products/jsolver

5. `BP(Box+Quad(rAI))`: a *Branch and Prune* solver which combines `Box`-consistency algorithm and the `Quad` algorithm where product terms are relaxed with the `rAI` heuristic

Note that the `BP(Box+simplex)` solver implements a strategy that is close to Yamamura's approach [31].

All the solvers have been parameterised to get solutions or boxes with a precision of 10^{-8}. That's to say, the width of the computed intervals is smaller than 10^{-8}. A solution is said to be *safe* if we can prove its existence and uniqueness within the considered box. This proof is based on the well known Brouwer fixpoint theorem (see [9]) and just requires a single test.
Columns $nSol$, $nSplit$ and $T(s)$ are respectively the number of found solutions, the number of branchings (or splittings) and the execution time in seconds. A "-" in the column $T(s)$ means that the solver was unable to find all the solutions within two hours. All the computations have been performed on a PC with Pentium IV processor at 2.66Ghz. The number of solutions is followed by the number of *safe* solutions between brackets.

Table 1 displays the performances of `RealPaver`, `BP(Box+Quad(Qmid))` and `BP(Box)`. The benchmarks have been grouped into three sets. The first group contains problems where the `Quad` solver does not behave very well. These problems are quite easy to solve with `Box`-consistency algorithm and the overhead of the relaxation and the call to a linear solver does not pay off. The second group contains a set of benchmarks for which the `Quad` solver compares well with the two other constraint solvers : the `Quad` solver requires always much less splitting and often less time than the other solvers. In the third group, which contains difficult problems, the `Quad` solver outperforms the two other constraint solvers. The latter were unable to solve most of these problems within two hours whereas the `Quad` solver managed to find all the solutions for all but two of them in less than 8 minutes.
For instance, `BP(Box)` requires about 74 hours to find the four solutions of the `Lee` benchmark whereas `Quad` managed to do the job in a couple of minutes. Likewise, the `Quad` solver managed to find forty safe solutions of the `stewgou40` benchmark in about 15 minutes whereas `BP(Box)` required about 400 hours.
The essential observation is that `Quad` solvers spend more time in the filtering step but they perform much less splitting than classical solvers. This strategy pays off for difficult problems.

Table 2 displays the performances of solvers combining `Box`-consistency and three different relaxation techniques. There is no significant difference between the solver based on the `Qmid` heuristics and the solver based on the `rAI` heuristics. Indeed, both heuristics provide convex and concave envelopes of the product terms.

The `Quad` solvers outperform Yamamura's approach for all benchmarks but `yama195`, which is a quasi-linear problem.

All the problems, except `cyclic5` and `reimer5`, contain many quadratic terms and some product and power terms. `cyclic5` is a pure multi-linear prob-

lem that contains only sums of products of variables. The `Quad` algorithm has not been very efficient for handling this problem. Of course, one could not expect an outstanding performance on this bench since product term relaxation is a poor approximation of multi-linear terms.

`reimer5` is a pure power problem of degree 6, that has been well solved by the `Quad` algorithm. Note that Verschelde's homotopy continuation machine [30] required about 10 minutes to solve this problem on Sparc Server 1000 and about 10 hours (on a PC equipped with a PII processor at 166Mhz) to solve `stewgou40`, another challenging problem. As opposed to the homotopy continuation method, the `Quad` solver is very simple to implement and to use. The performances on these difficult problems illustrate well the capabilities of the power relaxations.

5 Discussion

The approach introduced in this paper is related to some work done in the interval analysis community as well as to some work achieved in the optimisation community.

In the interval analysis community, Yamamura et. al. [31] have used a simple linear relaxation procedure where nonlinear terms are replaced by new variables to prove that some box does not contain solutions. No convex/concave outer-estimations are proposed to obtain a better approximation of the nonlinear terms. As pointed out by Yamamura, this approach is well adapted to quasi-linear problems : "*This test is much more powerful than the conventional test if the system of nonlinear equations consists of many linear terms and a relatively small number of nonlinear terms*" [31].

The global optimisation community worked also on solving nonlinear equation problems by transformation into an optimisation problem (see for example chapter 23 in [7]). The optimisation approach has the capability to take into account specific semantic of nonlinear terms by generating a tight outer-estimation of these terms. The pure optimisation methods are not rigorous since they do not take into account rounding errors and do not prove the existence and uniqueness of the solutions.

In this paper, we have exploited an RLT schema to take into account specific semantic of nonlinear terms. This relaxation process is incorporated in the *Branch and Prune* process [29] that exploits interval analysis and constraint satisfaction techniques to find all solutions in a given box. Experimental results show that this approach outperforms the classical constraint solvers.

A safe rounding process is a key issue for the `Quad` framework. Let's recall that the simplex algorithm is used to narrow the domain of each variable with respect to the subset of the linear set of constraints generated by the relaxation process. The point is that most implementations of the simplex algorithm are unsafe. Moreover, the coefficients of the generated linear constraints are computed with floating point numbers. So, two problems may occur in the `Quad`-filtering process:

1. The whole linearization may become incorrect due to rounding errors when computing the coefficients of the generated linear constraints ;

2. Some solutions may be lost when computing the bounds of the domains of the variables with the simplex algorithm.

We have proposed in [15] a safe procedure for computing the coefficients of the generated linear constraints. The second problem has been addressed by Neumaier [16]. He proposes a simple and cheap procedure to get a rigorous lower bound of the objective function. The incorporation of these procedures in the `Quad` framework will allow us to a safe use of the linear relaxations.

References

1. F.A. Al-Khayyal and J.E. Falk. Jointly constrained biconvex programming. *Mathematics of Operations Research*, pages 8:2:273–286, 1983.
2. C. Audet, P. Hansen, B. Jaumard, and G. Savard. Branch and cut algorithm for nonconvex quadratically constrained quadratic programming. *Mathematical Programming*, pages 87(1), 131–152, 2000.
3. F. Benhamou, D. McAllester, and P. Van-Hentenryck. CLP(intervals) revisited. In *Proceedings of the International Symposium on Logic Programming*, pages 124–138, 1994.
4. Y. Crama. Recognition problems for polynomial in 0-1 variables. *Mathematical Programming*, pages 44:139–155, 1989.
5. O. Didrit. *Analyse par intervalles pour l'automatique : résolution globale et garantie de problèmes non linéaires en robotique et en commande robuste*. PhD thesis, Université Parix XI Orsay, 1997.
6. Peter Dietmaier. The stewart-gough platform of general geometry can have 40 real postures. In *Advances in Robot Kinematics: Analysis and Control*, pages 1–10, 1998.
7. C.A. Floudas, editor. *Deterministic global optimization: theory, algorithms and applications*. Kluwer Academic Publishers, 2000.
8. Benhamou Frdric, Goualard Frdric, Granvilliers Laurent, and Puget Jean-Franois. Revising hull and box consistency. In *Proceedings of ICLP'99, The MIT Press*, pages 230–244, 1999.
9. Eldon R. Hansen. *Global Optimization Using Interval Analysis*. Marcel Dekker, New York, 1992.
10. H.Collavizza, F.Delobel, and M. Rueher. Comparing partial consistencies. *Reliable Computing*, pages Vol.5(3),213–228, 1999.
11. Yahia Lebbah, Michel Rueher, and Claude Michel. A global filtering algorithm for handling systems of quadratic equations and inequations. *Lecture Notes in Computer Science*, 2470:109–123, 2002.
12. T-Y Lee and J-K Shim. Elimination-based solution method for the forward kinematics of the general stewart-gough platform. In *In F.C. Park C.C. Iurascu, editor, Computational Kinematics, pages 259-267. 20-22 Mai*, 2001.
13. O. Lhomme. Consistency techniques for numeric csps. In *Proceedings of IJCAI'93*, pages 232–238, 1993.
14. A. Mackworth. Consistency in networks of relations. *Journal of Artificial Intelligence*, pages 8(1):99–118, 1977.
15. Claude Michel, Yahia Lebbah, and Michel Rueher. Safe embedding of the simplex algorithm in a csp framework. In *Proc. of 5th Int. Workshop on Integration of AI and OR techniques in Constraint Programming for Combinatorial Optimisation Problems CPAIOR 2003, CRT, Universit de Montral*, pages 210–220, 2003.

16. Arnold Neumaier and Oleg Shcherbina. Safe bounds in linear and mixed-integer programming. *Mathematical Programming, Ser. A*, pages 99:283–296, 2004.
17. A. Rikun. A convex envelope formula for multilinear functions. *Journal of Global Optimization*, pages 10:425–437, 1997.
18. G. Rote. The convergence rate of the sandwich algorithm for approximating convex functions. *Comput.*, pages 48:337–361, 1992.
19. H.S. Ryoo and V. Sahinidis. A branch-and-reduce approach to global optimization. *Journal of Global Optimization*, pages 8(2):107–139, 1996.
20. H.S. Ryoo and V. Sahinidis. Analysis of bounds for multilinear functions. *Journal of Global Optimization*, pages 19:403–424, 2001.
21. V. Sahinidis and M. Twarmalani. Baron 5.0 : Global optimisation of mixed-integer nonlinear programs. Technical report, University of Illinois at Urbana-Champaign, Department of Chemical and Biomolecular Engeneering, 2002.
22. V. Sahinidis and M. Twarmalani. Global optimization of mixed-integer programs : A theoretical and computational study. *Mathematical Programming, Ser. A*, pages 99:563–591, 2004.
23. H.D. Sherali. Convex envelopes of multilinear functions over a unit hypercube and over special discrete sets. *Acta mathematica vietnamica*, pages 22(1):245–270, 1997.
24. H.D. Sherali. Global optimization of nonconvex polynomial programming problems having rational exponents. *Journal of Global Optimization*, pages 12:267–283, 1998.
25. H.D. Sherali and C.H. Tuncbilek. A global optimization algorithm for polynomial using a reformulation-linearization technique. *Journal of Global Optimization*, pages 7, 1–31, 1992.
26. H.D. Sherali and C.H. Tuncbilek. A comparison of two reformulation-linearization technique based on linear programming relaxations for polynomial porgramming problems. *Journal of Global Optimization*, pages 10:381–390, 1997.
27. H.D. Sherali and C.H. Tuncbilek. New reformulation linearization/convexification relaxations for univariate and multivariate polynomial programming problems. *Operations Research Letters*, pages 21:1–9, 1997.
28. H.D. Sherali and H. Wang. Global optimization of nonconvex factorable programming problems. *Math. Program.*, pages 89:459–478, 2001.
29. P. Van-Hentenryck, D. Mc Allester, and D. Kapur. Solving polynomial systems using branch and prune approach. *SIAM Journal on Numerical Analysis*, pages 34(2):797–827, 1997.
30. J. Verschelde. The database of polynomial systems. Technical report, http://www.math.uic.edu/ jan/Demo/, 2003.
31. Kawata H. Yamamura K. and Tokue A. Interval solution of nonlinear equations using linear programming. *BIT*, pages 38(1):186–199, 1998.