# An interval partitioning algorithm for constraint satisfaction problems

## Chandra Sekhar Pedamallu*

New England Biolabs Inc.,
Ipswich, MA, 01938, USA
E-mail: pcs.murali@gmail.com
*Corresponding author

## Arun Kumar

School of Aerospace, Mechanical and Manufacturing Engineering,
RMIT University,
124 LaTrobe Street,
Melbourne, Victoria 3000, Australia
E-mail: a.kumar@rmit.edu.au

## Tibor Csendes

Department of Computational Optimisation,
University of Szeged,
H-6701 Szeged, P.O. Box 652, Hungary
E-mail: csendes@inf.u-szeged.hu

## Janos Posfai

New England Biolabs Inc.,
Ipswich, MA, 01938, USA
E-mail: posfai@neb.com

**Abstract:** We propose an efficient interval partitioning algorithm to solve the continuous constraint satisfaction problem (CSP). The method comprises a new dynamic tree search management system that also invokes local search in selected subintervals. This approach is compared with two classical tree search techniques and three other interval methods. We study some challenging kinematics problems for testing the algorithm. The goal in solving kinematics problems is to identify all real solutions of the system of equations defining the problem. In other words, it is desired to find all object positions and orientations that satisfy a coupled non-linear system of equations. The kinematics benchmarks used here arise in industrial applications.

**Keywords:** continuous constraint satisfaction problem; interval partitioning with local search; tree search strategies; kinematics problems.

**Reference** to this paper should be made as follows: Pedamallu, C.S., Kumar, A., Csendes. T. and Posfai, J. (2011) 'An interval partitioning algorithm for constraint satisfaction problems', *Int. J. Modelling, Identification and Control*, Vol. 14, Nos. 1/2, pp.133–140.

**Biographical notes:** Chandra Sekhar Pedamallu received his BTech in Mechanical Engineering from Nagarjuna University, India, in 1999, MTech in Industrial Management from Indian Institute of Technology Madras, India, in 2001 and PhD from the Division of Systems Engineering and Management, Nanyang Technological University, Singapore, in 2007. He worked as a Visiting Scientist and Project Leader in New England Biolabs Inc., USA and Indonet Global Limited (Division of Subuthi Overseas Inc., USA), India, respectively. He is currently working as a Postdoctoral Research Fellow in Bioinformatics at New England Biolabs, Inc., USA. His research interests are in the areas of bioinformatics, computational biology, global optimisation, parallel algorithms, interval-symbolic applications in non-linear programming and applications of meta-heuristics on OR problems. He is an editorial board member of three research journals and founding/active member of several European working groups on OR.

Arun Kumar is a Senior Lecturer of Manufacturing Engineering in the School of Aerospace, Mechanical and Manufacturing Engineering at RMIT University, Melbourne, Australia. He was previously employed as an Associate Professor of Systems and Engineering Management at Nanyang Technological University in Singapore and Associate Professor of Decision Sciences at

the State University of New Jersey. He obtained his PhD in Operations Research from Virginia Tech. His research interests are in the applications of probability to problems in reliability, logistics and healthcare systems. He is a member of INFORMS, IEEE, and European Simulation Society.

Tibor Csendes received his MSc and PhD in Mathematics from the Jzsef Attila University, Hungary, in 1980 and 1985; Candidate of Sciences and Doctor of Science from the Hungarian Academy of Sciences, Hungary, in 1993 and 2007. He is the Head of Department, Full Professor in the Institute of Informatics; and at present, the General and Scientific Vice Dean of the Faculty of Science and Informatics of the University of Szeged, Hungary. He is an expert in the areas of interval methods, global optimisation, and scientific computing. He has co-authored over 70 peer reviewed articles and some books, and edited several special issues of scientific journals. He is an editor of five research journals, and the President of the Hungarian Operations Research Society.

Janos Posfai earned his degrees in Mathematics and Physics in Hungary. Since 1987, he has been working on molecular biology related problems, applying the tools of mathematics and computer science in the analysis of DNA and amino acid sequences. His main interests are recognition of functional patterns in proteins and in DNA, identification and classification of restriction-modification systems, genome analysis and genomic data mining. He held various positions at EMBL Heidelberg, at University of California, San Francisco, at Cold Spring Harbor Laboratory, New York. Currently, he is a Senior Scientist of New England Biolabs, USA.

## 1 Introduction

Continuous constraint satisfaction problems (CSPs) can represent real world applications, ranging from chemical engineering to aircraft design. A CSP is defined by:

- a set of variables, $V = \{x_1, \ldots, x_n\}$
- a set of constraints, $C = \{c_1, \ldots, c_r\}$, for the variables $x_i$ of the problem: numeric constraints are linear or non-linear equations or inequalities. Let us denote them as

  $g_i(x_1, ..., x_n) \le 0 \; i = 1, ..., k,$

  $h_i(x_1, ..., x_n) = 0 \; i = k+1, ..., r,$

  and
- variable domains $D_i = [\underline{D_i}, \overline{D_i}]$ for $x_i$, $i = 1, \ldots, n$.

A solution of a CSP is an element ($x^*$) of the search space ($X = D_1 \times \cdots \times D_n$), that satisfies all equations and inequalities simultaneously.

We aim at identifying all real feasible solutions ($x^*$) for $C$ and propose an interval partitioning (IP) method to achieve this goal. Neumaier et al. (2005) compared the performance of major complete/global and incomplete/local solvers using an extensive set of CSPs. In particular, interval methods are complete because they never discard a sub-domain that possibly contains feasible solutions. This is a consequence of the proper use of interval arithmetic; see Alefeld and Herzberger (1983) for a complete description of interval extension, and Hansen (1992), Neumaier (1990), and Ratschek and Rokne (1995) for descriptions of interval methods. Conventionally, interval and non-interval partitioning methods use local search procedures to identify good feasible solutions. By principle, an IP method continues to subdivide a given box (sub-domain) until either it becomes infeasible and discarded by the cut-off test, or it becomes a small enclosure with the potential to contain a feasible solution. During the partitioning process, an increasing number of distinct feasible solutions are identified within promising boxes. A box is subject to local search only once, because after that it gets re-partitioned again and feasible solutions are searched in its child boxes. Therefore, it is not important whether or not a box contains multiple solutions, because the partitioning will continue until it is verified that the child box does not contain any other solutions. Lebbah (2003) described an interval method, interval constraints solver (ICOS) that is praised for its reliability in finding solutions of CSPs. However, a note is made on its slow convergence by Neumaier et al. (2005). Furthermore, results for ICOS are reported for the identification of a first feasible solution rather than for all feasible solutions.

An important issue in enhancing the convergence rate of an interval method is the selection of the subinterval to explore. This relates to the box ranking decision in the search tree. Here, we propose a new dynamic stage-wise tree search algorithm for box ranking. The latter approach also integrates a local search procedure feasible sequential quadratic programming, FSQP (Lawrence et al., 1997), in IP. We compare this approach to classical depth-first and best-first tree search techniques. Further, we propose a simple parallel variable domain bisection scheme to accelerate convergence. The minimum number of variables that are partitioned in parallel is two and the rule for bisecting the given subinterval along a variable is as follows. Every interval component whose calculated weight is more than the average weight of all candidate variables is bisected. These weights are determined according to certain criteria related to the number of constraints in the expression of which that the variable appears and the type of mathematical functions that it takes place. This partitioning scheme seems to be quite effective in reducing CPU time

and it makes IP a viable method for identifying all solutions of the CSP.

To evaluate the proposed IP, we conduct tests on a number of kinematics benchmarks published in COPRIN (2004). We compare the proposed tree management approach with the two classical strategies and also compare our results with ALIAS (COPRIN, 2004), a comprehensive set of C++ libraries that include 2B/3B box and hull consistency variants), ICOS (Lebbah, 2003) and QUAD [for two problems reported in Lebbah et al. (2003)]. ALIAS and QUAD results are provided for identifying all solutions, whereas ICOS results are reported for identifying a single solution.

## 2 The IP algorithm

An IP algorithm is basically a branch and bound procedure, and the domain $X$ is partitioned into smaller boxes $Y$ to be investigated. All such boxes $Y$ are placed in a pending list. When the leading box is taken from the pending list, the lower and upper bounds of each interval constraint are calculated using interval arithmetic. The following criteria hold for assessing the feasibility of a box $Y$.

- *Infeasible boxes*: When a constraint's interval bounds indicate infeasibility: If $\underline{G_i}(Y) > 0$ or $0 \notin H_i(Y)$ for any $i$, then box $Y$ is called an infeasible box and it is discarded. Here, $\underline{G_i}(Y)$ denotes the lower bound for the inclusion function of $g_i$ and $\overline{G_i}(Y)$ denotes its upper bound. $G_i(Y)$ denotes the range of the inclusion function for $g_i$. A similar notation holds for $h_i$.
- *Feasible boxes*: When all constraints are consistent over a given box $Y$, the box is deleted from the pending list and stored as a feasible box (or as a feasible enclosure of a small tolerance limit).
- *Indeterminate boxes*: When at least one constraint is *indeterminate*: If $(\underline{G_i}(Y) < 0$ and $\overline{G_i}(Y) > 0)$ or $0 \in H_i(Y) \neq 0$ for any $i$ and all other constraints are consistent or indeterminate, then box $Y$ is indeterminate.

In equations (1) and (2) we provide the definition of the uncertainty degree ($PG_Y^i$, or $PH_Y^i$) related to a constraint $i$ that is indeterminate over the box $Y$.

$$PG_Y^i = \overline{G_i}(Y), \tag{1}$$

and

$$PH_Y^i = \overline{H_i}(Y) + \left|\underline{H_i}(Y)\right|. \tag{2}$$

We also define the *total uncertainty degree* of a box, $IF_Y$, as the sum of the uncertainty degrees of equalities and inequalities that are indeterminate over $Y$.

Indeterminate boxes are partitioned by dividing at least two variables that have the largest width component intervals in the box. The resulting child boxes are placed in the pending list to be investigated. The search stops either when the theoretical number of solutions is identified (if it is known a priori) or when the pending list is exhausted. Below we provide the basic IP algorithm.

### *2.1 Basic IP procedure*

Step 0 Set the starting box $Y = X$. Set list of pending boxes, $B = \{Y\}$.

Step 1 If $B = \emptyset$, or if the a priori known number of solutions are found, then STOP and report all feasible solutions found. Else, pick the first box $Y \in B$.

- 1.1 If $Y$ is infeasible, remove $Y$ from $B$ and repeat Step 1.
- 1.2 If $Y$ is sufficiently small, store it as a possible feasible solution enclosure. Remove $Y$ from $B$ and repeat Step 1.

Step 2 Select directions to partition. Set $v$ to be the number of interval components to partition.

Step 3 Subdivide $Y$ into $2^v$ non-overlapping child boxes.

Step 4 Remove $Y$ from $B$, add the $2^v$ child boxes to $B$. Go to Step 1.

We want to identify those boxes in the pending list that contain consistent solutions while the branch and bound tree expands. Therefore, it is important to manage the pending list of boxes (i.e., the branch and bound tree) efficiently and investigate them in proper order so as to improve the convergence rate of the algorithm and discover as many feasible solutions as possible in the early stages of the search. Here, we propose a new tree management scheme and describe it as follows.

The tree management system in the proposed IP maintains a stage-wise branching scheme that is conceptually similar to the iterative deepening approach (Korf, 1985). The iterative deepening approach explores all nodes generated at a given tree level (stage) before it starts assessing the nodes at the next stage. Exploration of boxes at the same stage can be done in any order, the sweep may start from the best-first box, from the one on the most right, or most left of that stage. On the other hand, in the proposed adaptive tree management system, it is permitted to grow a sub-tree forming partial succeeding tree levels from a node (a parent box) at the current stage and to explore nodes in this sub-tree before exhausting the nodes at the current stage. That is, a box is selected from the children of the same parent with the maximal $IF_Y$ value as a criterion (this is actually the worst-first rule, but in classical terms we name it as best-first), and the child box is partitioned again continuing to build the same sub-tree. This sub-tree grows until the total area deleted (TAD) value fails to improve by discarding boxes in two consecutive partitioning iterations in this sub-tree.

Such failure triggers a call to local search where all boxes out of which no local search were started are processed by the procedure FSQP, after which they are placed back in the list of pending boxes. Exploration is then resumed among nodes at the current stage. Feasible solutions found by FSQP are stored. As mentioned previously, whether or not FSQP fails to find a solution IP continues to partition the box since it passes the cut-off test (test for feasibility) as long as it may contain a solution. Finally, the algorithm encloses potential solutions in sufficiently small boxes where local search can identify them. In IP, FSQP acts as a catalyst that occasionally scans larger boxes to identify solutions at earlier stages of the search. The proposed IP algorithm is described below.

### *2.2 IP with adaptive tree management*

Step 0 Set tree stage, $s = 1$, and future stage index, $r = 1$. Set non-improvement counter for *TAD*: $nc = 0$. Set $B_s$, the list of pending boxes at stage s be equal to $\{X\}$, and $B_{s+1} = \emptyset$.

Step 1 If $B_s = \emptyset$ and the current lower bound, CLB has not improved compared to the stage $s - 1$, or, both $B_s = \emptyset$ and $B_{s+1} = \emptyset$, then STOP.

Else, if $B_s = \emptyset$ and $B_{s+1} = \emptyset$, then set $s \leftarrow s + 1$, $r \leftarrow s$, and continue. Pick the first box $Y$ in $B_s$ and continue.

1.1 If $Y$ is infeasible or suboptimal, discard $Y$, and go to Step 1.

1.2 If $Y$ is sufficiently small, evaluate $m$, its midpoint, and if it is a feasible improving solution, update *CLB*, reset $nc \leftarrow 0$, and store $m$. Remove $Y$ from $B_s$ and go to Step 1.

Step 2 Select coordinate direction(s) along which partition is to be made (use the subdivision direction selection rule *IIR*). Set $v$ = number of variables to partition.

Step 3 Subdivide $Y$ into $2^v$ non-overlapping child boxes. Check *TAD*, if it improves, then reset $nc \leftarrow 0$, else set $nc \leftarrow nc + 1$.

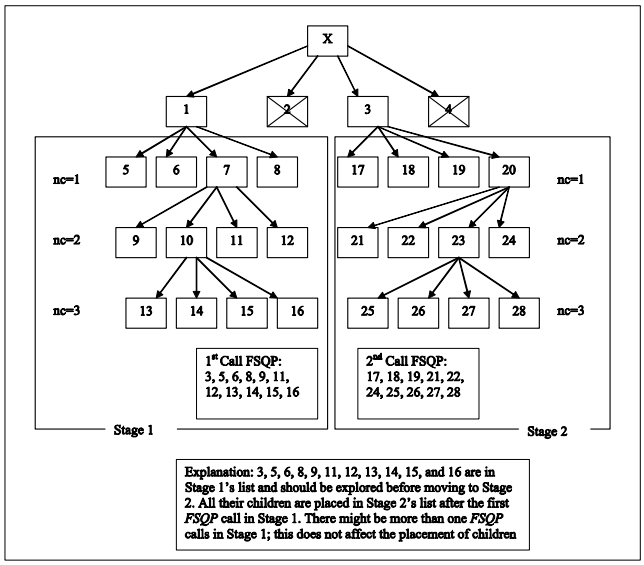
Step 4 Remove $Y$ from $B_s$, add the $2^v$ new boxes to $B_r$.

4.1 If $nc > 2$, apply FSQP to all (previously unprocessed by FSQP) boxes in $B_s$ and $B_{s+1}$, and reset $nc \leftarrow 0$. If FSQP is called for the first time in stage $s$, then set $r \leftarrow s + 1$.

4.2 Go to Step 1. The adaptive tree management system in *IP* is illustrated in Figure 1 on a small tree where node labels indicate the order of visiting the nodes.

As shown in the algorithm above, the adaptive tree management scheme maintains two lists of boxes, $B_s$ and $B_{s+1}$ that are the lists of boxes to be explored at the current stage $s$ and the next stage $s + 1$, respectively. Initially, the set of indeterminate boxes in the pending list $B_s$ consists only of $X$ and $B_{s+1}$ is empty. As child boxes of a selected parent box are added to a list, they are ordered in descending order of $IF_Y$. We choose this ranking strategy, because the algorithm aims to delete as many boxes as possible in the early stages of the search. Boxes in the sub-tree stemming from the selected parent at the current stage are explored and partitioned until there is no improvement in *TAD* in two consecutive partitioning iterations. At that point, partitioning of the selected parent box is stopped and all boxes that have not been processed by local search are sent to FSQP module and processed to identify feasible point solutions if possible. From that moment onwards, child boxes generated from any other selected parent in $B_s$ are stored in $B_{s+1}$ irrespective of further calls to FSQP in the current stage. When all boxes in $B_s$ have been assessed (discarded, stored as feasible boxes, or partitioned), the search moves to the next stage, $s + 1$, starting to explore the boxes stored in $B_{s+1}$. In this manner, a smaller number of boxes (those in the current stage) are maintained in primary memory and the search is allowed to go down to deeper levels within the same sub-tree, increasing the chances to discard boxes. On the other hand, by enabling the search to explore horizontally across boxes at the current stage, it might be possible to find feasible solutions faster by not partitioning parent boxes that are not so promising.

**Figure 1** Implementation of the adaptive iterative deepening procedure




The tree continues to grow in this manner taking up the list of boxes of the next stage after the current stage's list of boxes is exhausted. The algorithm stops when there are no more boxes in $B_s$ and $B_{s+1}$. As mentioned earlier, in case the exact number of solutions is known, the performance measure of the algorithm might then be the amount of CPU time taken to identify them all. Otherwise, the number of solutions found within limited CPU time would indicate the success of the algorithm.

## 3 Background of kinematics problems

### *3.1 Description of the problems*

The continuous CSP is a core topic in many real world engineering applications including kinematic analysis. Kinematics is fundamental in the design and control of robot manipulators (used particularly in contact analysis, assembly planning, position analysis, and path planning) since performance is achieved through the movement of links/legs whose geometry is crucial. Geometric kinematics calculates the state of a robot from measurements (direct kinematics) or poses (inverse kinematics), and provides answers for associated questions on accuracy and singularities. These problems require the identification of all object positions and orientations that satisfy a coupled non-linear system of equations (Dietmaier, 1998; Tsai and Morgan, 1985; Fu et al., 2006; Loudini et al., 2007). As a brief introduction, we provide a description of the inverse position problem for a six-revolute-joint problem. The inverse position problem for a six-revolute-joint problem in mechanics (referred to as test problem Kin2 later on) is illustrated in Figure 2.

**Figure 2** A general 6-R manipulator

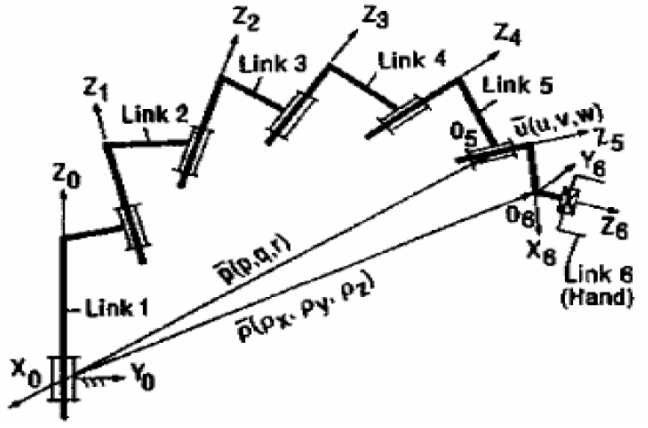
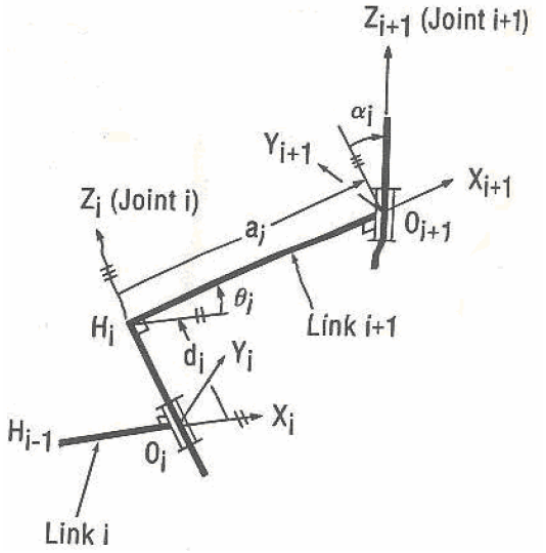



**Figure 3** The basic notation used in the optimisation problem




A 6R manipulator has six moving links, numbered sequentially from 2 to 6, as shown in Figure 2 (Tsai and Morgan, 1985). Link 1 is designated as the base (fixed to ground) and link 6 as the hand or the manipulator. Every two neighbouring links are connected by a joint that is associated with a joint axis $Z_i$, $i$ = 1 to 6. Let $Z_i$, and $Z_{i+1}$ be two adjacent joint axes and $H_iO_{i+1}$ be the directed common normal between $Z_i$, and $Z_{i+1}$. $H_i$ is the intersection of $H_iO_{i+1}$ and $Z_i$, and $O_{i+1}$ is the intersection of $H_iO_{i+1}$ and $Z_{i+1}$. Then one can define the following link parameters shown in Figure 3 (Hartenberg and Denavit, 1964):

$a_i$ The offset distance from the common normal $H_iO_{i+1}$.

$\alpha_i$ The angle to rotate the axis $Z_i$ about the common normal $H_iO_{i+1}$ so that $Z_i$ is parallel to $Z_{i+1}$. The sign of the rotation is given by the right hand screw rule with the screw taken along the normal $H_iO_{i+1}$.

$d_i$ The distance between the two normal $H_{i-1}O_i$ and $H_iO_{i+1}$ measured from $Z_i$. The sign of $d_i$ is positive if $O_iH_i$ points to the positive $Z_i$ direction. Otherwise, $d_i$ is negative.

$\Theta_i$ The angle with which to rotate the extended line of $H_{i-1}O_i$ about $Z_i$ so that the extended line of $H_{i-1}O_i$ is parallel to $H_iO_{i+1}$. The sign of rotation is given by the right hand screw with the screw pointing along the positive $Z_i$-axis.

If the $i^{th}$ joint is revolute, then $a_i$, $d_i$, and $i$ are constant while $i$ is variable. If the $i^{th}$ joint is prismatic, then $a_i$, $\alpha_i$, and $\Theta_i$ are constant while $d_i$ is a variable.

A coordinate system ($X_i$, $Y_i$, $Z_i$) is attached to each link of the manipulator as shown in Figure 2. In each coordinate system, the $Z_i$-axis is defined to align with the $i^{th}$ joint axis, the $X_i$-axis is the one along the extended line of $H_{i-1}O_i$; and the $Y_i$-axis is defined according to the right-hand screw rule. The first coordinate system is fixed to ground. Since the common normal $H_0O_1$ does not exist, the $X_1$-axis is chosen perpendicular to $Z_1$, in an arbitrary manner. Also, a seventh coordinate system is attached to the free-end to specify the position of the hand. The $Z_7$-axis lies in the direction from which the hand would approach an object as shown in Figure 2. The $X_7$-axis is defined by the common normal between the $Z_6$ and $Z_7$ axes, and the $Y_7$-axis is defined according to the right-hand screw rule again.

The equations representing the 6R problem are derived by first defining the coordinates of a point $P$ in the $i^{th}$ and $(i + 1)^{st}$ coordinate systems as $(p_{x_i}, p_{y_i}, p_{z_i})$ and $(p_{x_{i+1}}, p_{y_{i+1}}, p_{z_{i+1}})$, respectively. These two vectors are related to the hand position and orientation vectors by the equations of the form: $p_i = A_i p_{i+1}$, where $A_i$ is a matrix whose elements are $c_i = \cos\Theta_i$, $s_i = \sin\Theta_i$, $\lambda_i = \cos\ \alpha_i$, and $\mu_i = \sin\ \alpha_i$. The inverse transformation is written as: $p_{i+1} = A_i^{-1} p_i$.

By applying matrix transformation to each pair of coordinate systems between two successive links and proceeding from link 7 to link 1, the following equation is obtained: $p_1 = A_1A_2A_3A_4A_5A_6p_7$. Since an equivalent transformation matrix defines the relationship between the

coordinates of any point in the seventh system $p_7$, and that of the same point expressed in the first system, $p_1$, the matrix $A_{eq} = A_1A_2A_3A_4A_5A_6$ is known when the position and orientation of the hand is specified. Let $\rho$ ($\rho_x$, $\rho_y$, $\rho_z$) be the position vector from the origin of the first system to the origin of the seventh system as shown in Figure 2; and $l(l_x, l_y, l_z)$, $m(m_x, m_y, m_z)$ and $n(n_x, n_y, n_z)$ be three mutually perpendicular unit vectors aligned with the $X_7$, $Y_7$, and $Z_7$ axes, respectively. Then, when $\rho$, $l$, $m$, and $n$ are given in the first system, the equivalent $A$ matrix consists of the elements $\rho$, $l$, $m$, and $n$. By applying coordinate transformation and variable elimination, one can obtain a system of eight non-linear equations with eight unknowns expressed in the system of equations as given below (Tsai and Morgan, 1985), where $1 \le i \le 4$.

$$\begin{gathered} x_i^2 + x_{i+1}^2 - 1 = 0 \\ a_{1i}x_1x_3 + a_{2i}x_1x_4 + a_{3i}x_2x_3 + a_{4i}x_2x_4 \\ +a_{5i}x_5x_7 + a_{6i}x_5x_8 + a_{7i}x_6x_7 + a_{8i}x_6x_8 + a_{9i}x_1 \\ +a_{10i}x_2 + a_{11i}x_3 + a_{12i}x_4 + a_{13i}x_5 + a_{14i}x_6 + a_{15i}x_7 \\ +a_{16i}x_8 + a_{17i}x_8 = 0 \\ -1 \le x_i \le 1 \end{gathered} \tag{3}$$

The variables $x_i$ and $x_{i+1}$ represent the cosine ($c$) and sine ($s$) of the angle of the $i^{th}$ revolute joint to rotate (extended line of $H_{i-1}O_i$ about $Z_i$ so that the extended line $H_{i-1}O_i$ is parallel to $H_iO_{i+1}$) as illustrated in Figure 2. (The variables $x_1$, $x_2$, $x_3$, $x_4$, $x_5$, $x_6$, $x_7$, and $x_8$ actually represent the $c_1$, $s_1$, $c_2$, $s_2$, $c_4$, $s_4$, $c_5$, and $s_5$, respectively). In equation (1), the coefficients $a_{k_i}$ are defined as the manipulator parameters. For more details, we refer to Morgan and Sommese (1987).

## 4 Numerical evaluation

### 4.1 Test problems: kinematics applications

As stated before, kinematics is fundamental in the design and control of robot manipulators (used in contact analysis, assembly planning, position analysis, path planning) since performance is achieved through the movement of links/legs whose geometry is crucial. Geometric kinematics calculates the state of a robot from measurements (direct kinematics) or poses (inverse kinematics), and answers associated questions of accuracy and singularities. These problems require the identification of all object positions and orientations that satisfy a coupled non-linear system of equations (Dietmaier, 1998; Lebbah et al., 2003). Two main classes of interval-based methods have been applied in the robotics field: interval Newton (see Castellet, 1998; Merlet, 2001; Rao et al., 1998; Pedamallu, 2007); and box consistency techniques (Merlet, 2004; Van-Hentenryck et al. (1997a, 1997b). Both methods are used within the basic IP. Researchers have applied these methods to solve problems such as the 6R inverse kinematics problem, the direct kinematics of the Stewart-Gough platform, the general single-loop inverse kinematics and singularity analysis/mechanism design of parallel manipulators.

Six kinematics-robotics problems are used in the comparison, some of which are quite challenging. Here, we discuss some of their features. Direct kinematics (COPRIN, 2004) has two close solutions that are hard to isolate. This problem determines the pose parameters of a parallel robot platform and involves eight highly non-linear and inter-dependent trigonometric equations with three independent quadratic equations. Other difficult kinematics problems included here and not covered in COPRIN (2004) but solved by QUAD (Lebbah et al., 2003) are 6R (Kin2) and Stewart-Gough. Kin2 is a quadratic problem with ten real solutions and it describes the inverse position problem for six-revolute-joint. The Stewart-Gough is a manipulator configuration problem that has three independent constraints that might make constraint propagation based filtering methods such as 2B ineffective [as demonstrated in Lebbah et al., (2003)]. Numerica (described in Van-Hentenryck et al., 1997a), in which a box consistency technique is included) made more than 10,000,000 narrowing iterations to solve this problem. Another benchmark is the trigonometric Kin1-modified problem that describes the inverse kinematics of an elbow manipulator. Puma represents the inverse kinematics of a 3R robot whereas the KinCox is the simple inverse position problem. In Table 1, all test problems are listed with their details (number of dimensions, non-linear and linear equations, number of known feasible solutions), and providing their source references.

**Table 1** Characteristics of the kinematics test problems

| *Problem* | *Size* | *# of sol.* | *Source* |
|---|---|---|---|
| Kin2 | 8, 8, 0 | 10 | Lebbah et al. (2003) |
| | | | Van-Hentenryck et al. (1997a) |
| Kin1-modified | 6, 6, 0 | 16 | COPRIN (2004) |
| KinCox | 4, 4, 0 | 2 | COPRIN (2004) |
| Direct kinematics | 11, 11, 0 | 2 | COPRIN (2004) |
| Stewart-Gough | 9, 6, 3 | 2 | Lebbah et al. (2003) |
| Puma | 8, 7, 1 | 16 | COPRIN (2004) |

Notes: The size of the problem is expressed as the number of dimensions, the number of non-linear, and linear equations, respectively. The problems called Kin1-modified and direct kinematics are trigonometric, the others are quadratic.

### 4.2 Results

The impact of the adaptive tree management approach in IP is measured against depth-first and best-first branching strategies. The best-first strategy used here also ranks boxes according to maximum $IF_Y$ criterion, because IP aims to discard infeasible boxes as soon as possible. To allow a wider comparison, we also provide the published results of the following symbolic-interval methodologies: ALIAS (COPRIN, 2004), QUAD (Lebbah et al., 2003), and ICOS

(Lebbah, 2003) wherever results are available (Neumaier et al., 2005; Shcherbina et al., 2002). As mentioned previously, ALIAS is an extensive interval-symbolic software library where many of the local and global interval and symbolic filtering methods co-exist with special tools for univariate polynomials. QUAD is designed for filtering quadratic systems, its first stage involves linearisation, and the second stage uses a simplex algorithm to narrow the bounds on variables in the resulting linear programme. The authors show for their two illustrative examples (also included here) that QUAD is more efficient than the 2B and 3B consistency techniques and they compare their method with Numerica. ICOS is reported to be the most reliable method for the CSP among those compared by Neumaier et al. (2005), however, the published results are only for finding a single solution rather than all solutions.

We provide the summary of results in Table 2. In the first row, we convey the average CPU times necessary measured in standard time units. We restricted IP's run time by 1.31 standard time units [STU as defined in Shcherbina et al., 2002). One STU is equivalent to 318.369 seconds on our machine. All runs are executed on a PC with 256 MB RAM, 1.7 GHz P4 Intel CPU, on Windows platform. The IP code is developed with Visual C++ 6.0 interfaced with the PROFIL interval arithmetic library (Knüppel, 1994) and FSQP.

The second row shows the average number of tree stages in which the complete set of solutions is found. The third row conveys the number of FSQP calls. In the fourth row, we provide the average percentage of feasible solutions found with regard to the complete solution set. The fifth row indicates the average number of function calls. In the sixth row, we indicate the number of problems whose complete feasible solution set could not be identified within the given time limit. The seventh row shows the number of problems where the complete solution set is obtained within the shortest CPU time. From these results, it is observed that the proposed stage wise tree management strategy is quite effective as compared to other methods. Although the best-first strategy requires less function calls and FSQP calls, it is slower than the proposed adaptive method due its long pending list of boxes that need to be re-arranged at each update and also due to memory usage. The depth-first approach is quite inefficient in all respects as expected. ALIAS/QUAD had the best results for PUMA and Kin2. The reason for it is that these problems have ladder type constraints that are very suitable for constraint propagation, and the domain of one variable is reduced each time. For trigonometric expressions the filtering methods do not produce good results. It is interesting that QUAD, which is particularly developed for quadratic problems, is not as successful as IP solving the quadratic Stewart-Gough problem. ICOS is slower than ALIAS/QUAD in the four problems where it is comparable. When comparing IP with other interval methods it should be kept in mind that they all have a single common ground: the use of interval assessment. The proposed IP and these methods better complement each other rather than compete.

## 5 Conclusions

A generic collaborative methodology that integrates basic IP with local search (FSQP) is developed here to solve the CSP. The call for FSQP in IP is organised by an adaptive tree management system developed here. The latter system involves a stage-wise tree search: after the box to be partitioned is selected, the investigation is carried out only within the sub-tree under the selected parent box until it is decided that the infeasible area discarded does not improve in a number of consecutive iterations. At that point, all boxes under the sub-tree are sent as a batch to FSQP. The search then goes back to unexplored boxes in the current stage. In this sense, the depth of the sub-tree is bounded by the area removal performance criterion and its width is restricted by the number of parallel boxes in the sub-tree that grows under the selected box.

**Table 2** Summary of results

| *Indicator* | *Adapative* | *Best-first* | *Depth-first* | *ALIAS/ QUAD* | *ICOS* |
|---|---|---|---|---|---|
| CPU | 0.169 | 0.423 | 0.664 | 0.437 | 0.705 |
| NTS | 1.667 | - | - | - | - |
| NF | 852.83 | 241.83 | 2,021.50 | - | - |
| PSF | 100.00 | 96.00 | 65.00 | - | - |
| NFE | 20,540 | 3,159 | 316,609 | - | - |
| NPF | 0 | 1 | 3 | 1 | 2 |
| NMT | 5 | 0 | 0 | 3 | 0 |
| NUP | 0 | 0 | 0 | 0 | 2 |

Notes: CPU stands for the average CPU time needed (in STU), NTS for the average number of tree stages, NF for the average number of FSQP calls, PSF for the average percents of cases when the solution has been found, NFE for the average number of function evaluations, NPF for the number of problems with all feasible solutions found within the given time limit, NMT for the number of problems solved with the shortest CPU time, and NUP for number of reported unsolved problems.

Computational test results on a number of kinematics benchmarks show that the collaborative method is able to converge faster than best-first and depth-first tree search strategies. The proposed IP seems to be a promising approach with a potential for further improvement by the application of advanced symbolic-interval techniques such as the ones used in ALIAS. The presented methodology can be used efficiently for solving constrained optimisation problems (see also Pedamallu et al., 2007a, 2007b).

## Acknowledgements

We wish to thank Professor Andre Tits (Electrical Engineering and the Institute for Systems Research, University of Maryland, USA) for providing the source code of CFSQP. Also, we wish to thank Professor Linet Ozdamar and Dr. Tamás Vinkó for their valuable comments and suggestions.

## References


Alefeld, G. and Herzberger, J. (1983) *Introduction to Interval Computations*, Academic Press Inc., New York, USA.

Castellet, A. (1998) 'Solving inverse kinematics problems using an interval method', PhD Thesis, Universitat Politechnica de Catalunya, Barcelona, Spain.

COPRIN (2004) Available at http://www-sop.inria.fr/coprin/logiciels/ALIAS/Benches/.

Dietmaier, P. (1998) 'The Stewart-Gough platform of general geometry can have 40 real postures', *Proceedings of ARK*, pp.7–16, Strobl, Austria.

Fu, S., Yao, Y., and Shan, T. (2006) 'Non-linear robust control with partial inverse dynamic compensation for a Stewart platform manipulator', *International Journal of Modelling, Identification and Control*, Vol. 1, No. 1, pp.44–51.

Hansen, E. (1992) *Global Optimization Using Interval Analysis*, Marcel Dekker, New York, USA.

Hartenberg, R.S. and Denavit, J. (1964) *Kinematic Synthesis of Linkages*, McGraw-Hill, New York, USA.

Knüppel, O. (1994) 'PROFIL/BIAS – a fast interval library', *Computing*, Vol. 53, Nos. 3–4, pp.277–287.

Korf, R.E. (1985) 'Depth-first iterative deepening: an optimal admissible tree search', *Artificial Intelligence*, Vol. 27, No. 1, pp.97–109.

Lawrence, C.T., Zhou, J.L. and Tits, A.L. (1997) 'User's guide for CFSQP version 2.5: a code for solving (large scale) constrained nonlinear (minimax) optimization problems, generating iterates satisfying all inequality constraints', Institute for Systems Research, University of Maryland, College Park, MD.

Lebbah, Y. (2003) 'ICOS (interval constraints solver)', WWW-document, available at http://sites.google.com/site/ylebbah/icos.

Lebbah, Y., Rueher, M. and Michel, C. (2003) 'A global filtering algorithm for handling systems of quadratic equations and inequalities', *Lecture Notes Computer Science*, Vol. 2470, pp.109–213.

Loudini, M., Boukhetala, D. and Tadjine, M. (2007) 'Comprehensive mathematical modelling of a lightweight flexible link robot manipulator', *Int. J. of Modelling, Identification and Control*, Vol. 2, No. 4, pp.313–321.

Merlet, J-P. (2001) 'An improved design algorithm based on interval analysis for parallel manipulator with specified workspace', *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pp.1289–1294, Seoul, South Korea.

Merlet, J-P. (2004) 'Solving the forward kinematics of a Gough type parallel manipulator with interval analysis', *The International Journal of Robotics Research*, Vol. 23, No. 3, pp.221–235.

Morgan, A.P. and Sommese, A. (1987) 'Computing all solutions to polynomial systems using homotopy continuation', *Appl. Math. Comput*., Vol. 24, No. 2, pp.115–138.

Neumaier, A. (1990) *Interval Methods for Systems of Equations Encyclopedia of Mathematics and its Applications*, Cambridge University Press, Cambridge.

Neumaier, A., Shcherbina, O., Huyer W. and Vinkó, T. (2005) 'A comparison of complete global optimization solvers', *Mathematical Programming*, Vol. 103, No. 2, pp.335–356.

Pedamallu, C.S. (2007) 'New interval partitioning algorithms for global optimization problems', PhD thesis, Nanyang Technological University, Singapore.

Pedamallu, C.S., Ozdamar, L. and Csendes, T. (2007a) 'An interval partitioning approach for continuous constrained optimization', in *Models and Algorithms in Global Optimization*, Springer, USA.

Pedamallu, C.S., Ozdamar, L. and Csendes, T. (2007b) 'Symbolic interval inference approach for subdivision direction selection in interval partitioning algorithms', *J. Global Optimization*, Vol. 37, No. 2, pp.177–194.

Rao, R.S., Asaithambi, A. and Agrawal, S.K. (1998) 'Inverse kinematic solution of robot manipulators using interval analysis', *ASME Journal of Mechanical Design*, Vol. 120, No. 1, pp.147–150.

Ratschek, H. and Rokne, J. (1995) 'Interval methods', in Horst, R. and Pardalos, P.M. (Eds.): *Handbook of Global Optimization*, pp.751–828, Kluwer Academic Publisher, The Netherlands.

Shcherbina, O., Neumaier, A., Sam-Haroud, D., Vu, X-H. and Nguyen, T-V. (2002) 'Benchmarking global optimization and constraint satisfaction codes', *Global Optimization and Constraint Satisfaction: First International Workshop on Global Constraint Optimization and Constraint Satisfaction*, Valbonne-Sophia Antipolis, France.

Tsai, L.W. and Morgan, A.P. (1985) 'Solving the kinematics of the most general six and five-degree-of-freedom manipulators by continuation methods', *Journal of Mechanisms, Transmissions, and Automation in Design*, Vol. 107, No. 18, pp.189–200.

Van-Hentenryck, P., Michel, L. and Deville, Y. (1997a) *Numerica: a Modeling Language for Global Optimization*, MIT Press, London, England.

Van-Hentenryck, P., McAllester, D. and Kapur, D. (1997b) 'Solving polynomial systems using branch and prune approach', *SIAM Journal on Numerical Analysis*, Vol. 34, No. 2, pp.797–827.