Robust solutions for a robotic manipulator optimization problem

Ricardo Soto, Stéphane Caro, Broderick Crawford, Eric Monfroy

Robust Solutions for a Robotic Manipulator Optimization Problem

Ricardo Soto[1], Stéphane Caro[2], Broderick Crawford[1], and Eric Monfroy[3]

[1] Pontificia Universidad Católica de Valparaíso, Chile
[2] IRCCyN, Ecole Centrale de Nantes, France
[3] CNRS, LINA, Université de Nantes, France
{ricardo.soto,broderick.crawford}@ucv.cl
stephane.caro@irccyn.ec-nantes.fr
eric.monfroy@univ-nantes.fr

Abstract. In robotics, pose errors are known as positional and rotational errors of a given mechanical system. Those errors are commonly produced by the so-called joint clearances, which are the play between pairing elements. Predicting pose errors can be done via the formulation of two optimization models holding continuous domains, which belong to the NP-Hard class of problems. In this paper, we focus on the use of constraint programming in order to provide rigorous and reliable solution to this problem.

1 Introduction

Accuracy is one of the key features that favor robotic manipulators for many industrial applications. Superior levels of accuracy are achieved by controlling or measuring all possible sources of errors on the pose of the moving platform of a robotic manipulator. Joint clearance is one of most important sources of errors. It introduces extraneous degrees of freedom between two connected links. When present, they generally contribute importantly to the degradation of the performance of a mechanism. Various approaches have been proposed to compute and quantify the errors due to joint clearances [18, 10, 14, 6, 16, 15, 12], however none of them focuses on the reliability of solutions, which is mandatory to provide an accurate prediction of the pose error.

In this paper, we investigate the use of constraint programming in conjunction with interval analysis with the aim of guaranteeing the reliability of solutions. To this end, we combine a branch and bound algorithm with interval analysis, which allow drawing firm bounds on the pose errors given possible ranges for the clearances. We illustrate experimental results where the proposed approach generally outperforms the well-known solvers GAMS/BARON [1] and Ecl^{i}ps^e [17], while providing reliable solutions.

The remaining of this paper is organized as follows. Section 2 presents an error-prediction model used to characterize the influence of joint clearances on the end-effector pose of any robotic mechanical system. An overview of constraint programming —including the implemented approach— is presented in Sect. 3.

The experiments are presented in Sect. 4, followed by the conclusions and future works.

2 The Problem Formulation

In order to generalize the application of our approach, in this paper we consider robotic mechanical systems with the following configuration: n revolute joints, n links and an end-effector. Hence, the manipulator is assumed to be composed of $n+1$ links and n joints; where 0 is the fixed base, while link n is the end-effector. Next, a coordinate frame $\mathcal{F}_j$ is defined with origin O_j and axes X_j, Y_j, Z_j. This frame is attached to the $(j-1)$st link for $j = 1, \ldots, n+1$.

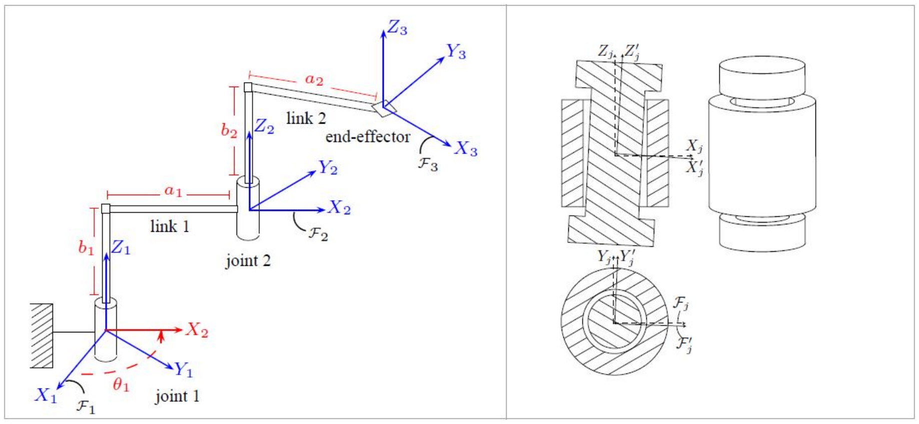

Fig. 1. Left: A serial manipulator composed of two revolute joints. Right: Clearance-affected revolute joint.

Figure 1 depicts an instance of such system considering $n = 2$ involving joints with join clearance as illustrated on the right side of Figure 1. Our goal is to find the maximal positional (translational) and rotational error for a given robotic configuration. Let $\delta \mathbf{p}_r$ and $\delta \mathbf{p}_t$ representing the rotational and translational errors of the manipulator end-effector, respectively. Then, the maximal pose error of the end-effector for a given manipulator configuration can be obtained by solving the following two optimization problems:

$$\begin{aligned} \text{maximize} \quad & \delta p_r, \\ \text{subject to} \quad & \delta r_{j,X}^2 + \delta r_{j,Y}^2 - \Delta\beta_{j,XY}^2 \leq 0, \\ & \delta r_{j,Z}^2 - \Delta\beta_{j,Z}^2 \leq 0, \\ & j = 1, \ldots, n \end{aligned} \tag{1}$$

$$
\begin{aligned}
\text{maximize} \quad & \delta p_t, \\
\text{subject to} \quad & \delta r_{j,X}^2 + \delta r_{j,Y}^2 - \Delta\beta_{j,XY}^2 \le 0, \\
& \delta r_{j,Z}^2 - \Delta\beta_{j,Z}^2 \le 0, \\
& \delta t_{j,X}^2 + \delta t_{j,Y}^2 - \Delta\gamma_{j,XY}^2 \le 0, \\
& \delta t_{j,Z}^2 - \Delta\gamma_{j,Z}^2 \le 0, \\
& j = 1, \dots, n
\end{aligned}
\tag{2}
$$

where $\delta r_{a,b}$ corresponds to the small rotation in joint a with respect to axis b, and $\delta t_{a,b}$ corresponds to the translation in joint a with respect to axis b. β and γ are simply constants used to limit the pose errors depending on the given configuration (an extended explanation of this model can be found in [4]). Let us notice that the two problems are nonconvex quadratically constrained quadratic (QCQPs). Indeed, all constraints of both problems are convex, their feasible sets are convex, and their objectives functions are non-convex. Then, both problems belong to the category of NP-Hard problems [10].

3 Constraint Programming for Global Optimization

3.1 Definitions

A Constraint Satisfaction Problem (CSP) is a formal problem representation, which mainly consists in a sequence of variables holding a domain and a set of relations over those variables called constraints. The idea is to find values for those variables so as to satisfy the constraints. The software technology devoted to tackle this problem is named Constraint Programming (CP). In this work, due to the presence of continuous decision variables, we focus on Numerical CSP (NCSP), which is an extension of a CSP devoted to continuous domains. Formally, a NCSP $\mathcal{P}$ is defined by a triple $\mathcal{P} = \langle \mathbf{x}, [\mathbf{x}], \mathcal{C} \rangle$ where:

- $\mathbf{x}$ is a finite sequence of variables $(x_1, x_2, \dots, x_n)$.
- $[\mathbf{x}]$ is a finite set of real intervals $([x_1], [x_2], \dots, [x_n])$ such that $[x_i]$ is the domain of x_i.
- $\mathcal{C}$ is a set of constraints $\{c_1, c_2, \dots, c_m\}$.

A solution to a NCSP is a set of real intervals that satisfy all the constraints. Optimization problems are handled in the same way. Hence, the 4-tuple $\mathcal{P} = \langle \mathbf{x}, [\mathbf{x}], \mathcal{C}, f(x) \rangle$ is employed in this case, where $f(x)$ is the cost function to be maximized or minimized.

3.2 NCSP Solving

In order to guarantee accurate solutions, NCSPs cannot be handled in the same way that CSPs mainly due to the presence of constraints over real numbers. Indeed, the representation of reals in numerical computations is not exact since it is commonly done by means of floating-point numbers, which are a finite set of rational numbers. This inaccuracy may lead to rounding errors and as a consequence to reach wrong solutions. One solution for rigorously dealing with real numbers relies on the integration of interval analysis on the solving process. The idea is to compute approximations over domains represented by intervals bounded by floating-point numbers [3]. A detailed presentation of interval analysis can be seen in [13], and some examples devoted to robotics in [7].

Then, the core idea for solving NCSPs relies in combining a branch and prune algorithm with interval analysis for handling continuous domains. A tree-data structure that holds intervals as the potential solutions is built on the fly by interleaving branching and pruning phases. The branching phase is responsible for creating the branches of the tree by splitting real intervals, while the pruning tries to filter from domain intervals that do not conduce to any feasible solution. The idea is to speed-up the solving process. This is possible by applying consistency techniques for continuous domains such as the hull and the box consistency [9, 2], which are similar to the arc-consistency [11] for finite domain CSPs.

Figure 2 depicts an algorithm for rigorously handling NCSPs. The procedure begins by receiving as input the set of constraint and domains of the problem. Then, four actions are embedded in a while loop. The **Contract** operator is responsible for pruning the tree, and **Split** applies a dichotomic division of intervals in order to carry out the branching process. Every computation of elementary operations $\{+, -, \times, /\}$ is done by using interval arithmetic. The process stops when the real values of the solution have reached the precision required of the problem.

```
Algorithm 1
Input: $\mathcal{C} = \{c_1, \ldots, c_m\}$, $[\mathbf{x}]$
1 $\mathcal{L} \leftarrow \{[\mathbf{x}]\}$
2 While $\mathcal{L} \neq \emptyset$ and $\neg$stop_criteria do
3     $([\mathbf{x}], \mathcal{L}) \leftarrow$ Extract($\mathcal{L}$)
4     $[\mathbf{x}] \leftarrow \mathbf{Contract}_{\mathcal{C}}([\mathbf{x}])$
5     $\{[\mathbf{x}'], [\mathbf{x}'']\} \leftarrow$ Split($[\mathbf{x}]$)
6     $\mathcal{L} = \mathcal{L} \cup \{[\mathbf{x}'], [\mathbf{x}'']\}$
7 End While
8 Return($\mathcal{L}$)
```

```
Algorithm 2
Input: $f$, $\mathcal{C} = \{c_1, \ldots, c_m\}$, $[\mathbf{x}]$
1  $\mathcal{L} \leftarrow \{[\mathbf{x}]\}$
2  $m \leftarrow +\infty$
3  While $\mathcal{L} \neq \emptyset$ and $\neg$stop_criteria do
4      $([\mathbf{x}], \mathcal{L}) \leftarrow$ Extract($\mathcal{L}$)
5      $[\mathbf{x}] \leftarrow \mathbf{Contract}_{\mathcal{C} \cup \{f(\mathbf{x}) \leq m\}}([\mathbf{x}])$
6      $m \leftarrow$ Update($[\mathbf{x}], f$)
7      $\{[\mathbf{x}'], [\mathbf{x}'']\} \leftarrow$ Split($[\mathbf{x}]$)
8      $\mathcal{L} = \mathcal{L} \cup \{[\mathbf{x}'], [\mathbf{x}'']\}$
9  End While
10 Return($\mathcal{L}, m$)
```

Fig. 2. Left: The branch and prune algorithm. Right: The branch and bound algorithm

A slight modification to the previous algorithm is required to handle optimization problems. Indeed, here a CP-based branch and bound algorithm is combined with interval analysis (see Figure 2). The corresponding cost function f has been added to the input set. A variable m is initialized to $+\infty$ in order to maintain an upper bound on the global minimum. In this way, potential solutions exceeding this bound are discarded by adding to the set of constraints. Five instructions are embedded in the same while loop. Now, **Contract** takes into account the cost function in order to prune the tree. The **Update** function has been added for updating the upper bound once better solutions are found. The branch and bound algorithm described above has been implemented on top of the RealPaver solver [5]. This implementation is used for the experiments presented in the next section.

4 Experiments

As previously mentioned the RealPaver solver has been used as base for our branch and bound algorithm. In this section, we verify the reliability of solutions and we compare the performance of such implementation with the state-of-the-art solvers GAMS/BARON [1] and Ecl^{i}ps^{e} [17]. Ecl^{i}ps^{e} is a widely used solver in CP and one of the few having support for continuous domains. GAMS/BARON is a popular solver from the mathematical programming field devoted to particularly hard optimization problems.

The following experiments take into account 4 rotation and 4 translation models (see Table 1). We consider from two to five joints for each model. The information related to the problem size (number of variables, number of constraints, and number of operators) is also given. Then, we provide the solving times in milliseconds for BARON; Ecl^{i}ps^{e}, and for the proposed algorithm. The experiments were run on a 3 Ghz Intel Pentium D Processor 925 with 2 GB of RAM running Ubuntu 9.04. All solving times are the best of ten runs.

The results show that our implementation is in general faster than its competitors. For smaller problems involving two and three joints, our algorithm exhibits excellent performance, being 100 times faster that BARON. This performance is clearly reached by the efficient filtering work done by the HC4. This is also influenced by the fact that there is no need for highly accurate computations for this particular problem: we actually do not have to reach the usual precision, which makes the search process less costly than they usually are.

In the presence of rotational problems considering four and five joints, our approach remains faster. However, once the number of joints increases, in particular when the number of variables is greater than 20, the searching begins to be slow. This is obviously explained by the exponential complexity in the problem size that leads to complete search methods like our branch and bound to a slower convergence. Despite of this common phenomenon, we estimate that the presented solving times are reasonable regarding the problem complexity as well as the solving techniques involved.

Table 1. Type, problem size and solving times (in seconds).

#joints	Type	Problem Size			BARON	RealPaver	Ecl^{i}ps^e
		#var	#ctr	#op			
2	T	12	8	28	0.08	0.004	>60
2	R	6	4	18	0.124	0.004	>60
3	T	18	12	135	0.124	0.008	t.o.
3	R	9	6	90	0.952	0.004	t.o.
4	T	24	16	374	0.144	0.152	t.o.
4	R	12	8	205	2.584	0.02	t.o.
5	T	30	20	1073	0.708	3.71	t.o.
5	R	15	10	480	9.241	0.26	t.o.

Let us remark that although RealPaver as well as Ecl^{i}ps^e use CP-based solving techniques. The performance of our approach is significantly better, this is not surprising due to RealPaver was originally designed for tackling problems with continuous domains.

Besides, solving times may actually not be the only point to emphasize here: It may also be important to discuss the reliability of the solutions given by BARON. We could indeed verify using RealPaver that the optimum enclosure computed by BARON is unfeasible on some of the above tested problem instances. It was already noted in [8] that BARON may not always be reliable, where the following example illustrates how BARON can fail to give the correct value and return a an inconsistent point instead. Let x, y be two real variables, $x, y \in [-10, 10]$, and two constraints $y - x^2 \geq 0$ and $y - x^2(x-2) + 10^{-5} \leq 0$. When BARON tries to find the lowest x for which both constraints are satisfied, it returns the point $(0, 0)$ although it is inconsistent w.r.t. the constraints of the problem (cf Figure 3, reproduced from [8]).

Finally, let us focus on the results obtained for the manipulator composed of two revolute joints illustrated in Fig. 1. Let us assume its geometric parameters are defined as follows:

$$
\begin{aligned}
a_1 &= 1 \text{ m} \\
b_1 &= 0 \text{ m} \\
\alpha_1 &= 0 \text{ rad} \\
a_2 &= 0.7 \text{ m} \\
b_2 &= 0 \text{ m} \\
\alpha_2 &= 0 \text{ rad}
\end{aligned}
$$

and the joint clearances are equal to:

$$\begin{aligned}\Delta\beta_{j,XY} &= 0.01 \text{ rad}\\ \Delta\beta_{j,Z} &= 0.01 \text{ rad}\\ \Delta b_{j,XY} &= 2 \text{ mm}\\ \Delta b_{j,Z} &= 2 \text{ mm}\end{aligned}$$

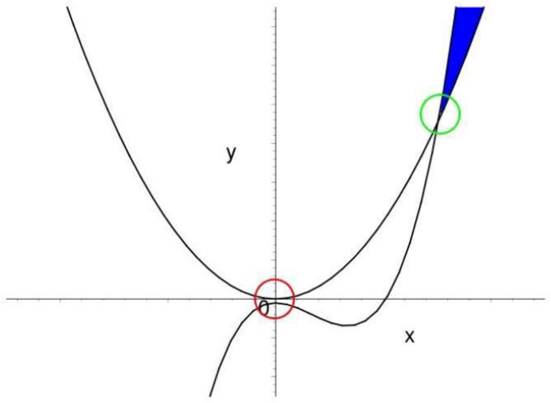

Fig. 3. BARON fails to find the global minimum and returns an inconsistent point instead.

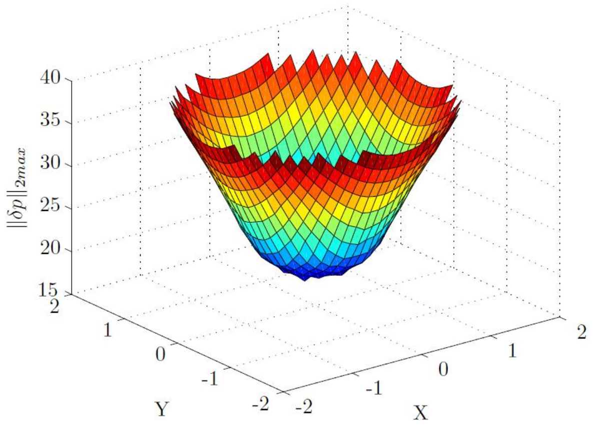

Fig. 4. 3D plot of the maximum positioning error of the manipulator with two joints throughout its Cartesian workspace.

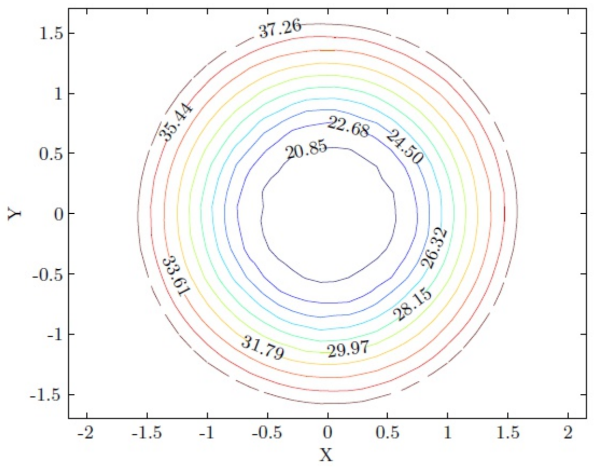

Fig. 5. Isocontours of the maximum positioning error in millimeters of the manipulator with two joints throughout its Cartesian workspace obtained with RealPaver

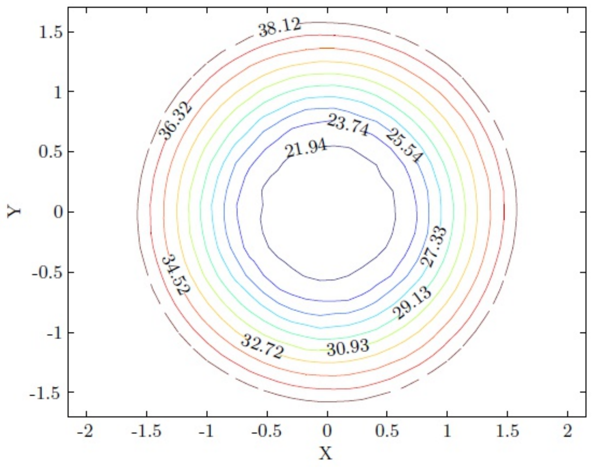

Fig. 6. Isocontours of the maximum positioning error in millimeters of the manipulator with two joints throughout its Cartesian workspace obtained with BARON

RealPaver and BARON were used to solve the optimization problem for all manipulator poses. Figures 5 and 6 show the value of the maximum positioning error of the end-effector obtained with RealPaver and BARON, respectively. This value is expressed in millimeters and plotted throughout the manipulator workspace. We can notice that the maximum obtained with BARON is always higher than the one obtained with RealPaver. The difference in any point between the results given by the two solvers is around one millimeter and decreases when positioning error rises: As we already mentioned it, we could notice that the solutions obtained with BARON may not be feasible.

5 Conclusions and Future Works

In this paper we have modeled and solved the complex problem of predicting the maximum pose error in robotic mechanical systems. To this end we have employed constraint programming techniques and interval analysis. Experimental results demonstrate the effectiveness of our approach where it is able to in general outperform well-known solvers such as BARON and Ecl^{i}ps^e, providing reliable solutions. Let us also mention that the presented approach is not only devoted to robotics, it in fact applies to whatever problem requiring rigorous computation and reliability in the solutions.

In this context, there are several directions for future work. In the future we plan to design new pruning criteria for accelerating the convergence of solutions. In this way, it would be easier to handle the exponential growth of the space of solutions. Finally, it would be interesting to pay an extended attention to the unreliability of solutions obtained by BARON. Such a problem could be also present in additional solvers.

References

1. GAMS. http://www.gams.com/ (Visited 10/2009).
2. F. Benhamou, D. Mc Allester, and P. Van Hentenryck. CLP(Intervals) Revisited. In *Proceedings of ILPS*, pages 124–138. MIT Press Cambridge, MA, USA, 1994.
3. F. Benhamou and W.J. Older. Applying Interval Arithmetic to Real, Integer and Boolean Constraints. *Journal of Logic Programming*, 32(1):1–24, 1997.
4. N. Binaud, P. Cardou, S. Caro, and P.Wenger. The kinematic sensitivity of robotic manipulators to joint clearances. In *Proceedings of ASME Design Engineering Technical Con- ferences, August 15-18*, Montreal, Canada, 2010.
5. L. Granvilliers and F. Benhamou. Algorithm 852: RealPaver: an Interval Solver Using Constraint Satisfaction Techniques. *ACM Trans. Math. Softw.*, 32(1):138–156, 2006.
6. C. Innocenti. Kinematic clearance sensitivity analysis of spatial structures with revolute joints. *ASME Journal of Mechanical Design*, 124(1):52–57, 2002.
7. 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, 2001.

8. Y. Lebbah, C. Michel, and M. Rueher. An Efficient and Safe Framework for Solving Optimization Problems. *Journal of Computational and Applied Mathematics*, 199(2):372–377, 2007. Special Issue on Scientific Computing, Computer Arithmetic, and Validated Numerics (SCAN 2004).
9. O. Lhomme. Consistency Techniques for Numeric CSPs. In *Proceedings of IJCAI*, pages 232–238, 1993.
10. P. D. Lin and J. F. Chen. Accuracy analysis of planar linkages by the matrix method. *Mechanism and Machine Theory*, 27(5):507–516, 1992.
11. A. Mackworth. Consistency in Networks of Relations. *Artificial Intelligence*, 8(1):99–118, 1977.
12. J. Meng, D. Zhang, and Z. Li. Accuracy analysis of parallel manipulators with joint clearance. *ASME Journal of Mechanical Design*, 131(1):011013–1–011013–9, 2009.
13. R. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs N. J., 1966.
14. J. Zhu K.-L. Ting. Uncertainty analysis of planar and spatial robots with joint clearances. *Mechanism and Machine Theory*, 35(9):1239–1256, 2000.
15. S. Venanzi and V. Parenti-Castelli. A new technique for clearance influence analysis in spatial mechanisms. *ASME Journal of Mechanical Design*, 127(3):446–455, 2005.
16. P. Voglewede and I. Ebert-Uphoff. Application of workspace generation techniques to determine the unconstrained motion of parallel manipulators. *ASME Journal of Mechanical Design*, 126(2):283–290, 2004.
17. M. Wallace, S. Novello, and J. Schimpf. ECLiPSe: A Platform for Constraint Logic Programming. Technical report, IC-Parc, Imperial College, London, 1997.
18. H. H. S. Wang and B. Roth. Position errors due to clearances in journal bearings. *ASME Journal of Mechanisms, Transmissions, and Automation in Design*, 111(3):315–320, 1989.