Using biased randomization for trajectory optimization in robotic manipulators

Alba Agustín[1], Alberto Olivares[2], Ernesto Staffetti[2]

[1] Dept. of Statistics and OR, Public University of Navarre,
Los Magnolios Blg. Campus Arrosadia
31006 Pamplona, Spain
albamaria.agustin@unavarra.es

[2] Dept. of Information and Communication Technology, Rey Juan Carlos University,
Camino del Molino, s/n
28943 Fuenlabrada, Spain
{alberto.olivares, ernesto.staffetti}@urjc.es

Abstract. We study the problem of optimization of trajectories for a robotic manipulator, with two degrees of freedom, which is constrained to pass through a set of waypoints in the workspace. The aim is to determine the optimal sequence of points and continuous optimal system trajectory. The actual formulation involves an optimal control problem of a dynamic system within integer variables that model the waypoints constrains. The nature of this problem, highly nonlinear and combinatorial, makes it particularly difficult to solve. The proposed method combines a meta-heuristic algorithm to determine the promising sequence of discrete points with a collocation technique to optimize the continuous path of the system. This method does not guarantee the global optimum, but can solve instances of dozens of points in reasonable computation time.

Keywords: robotics, optimal control, motion planning, meta-heuristics, biased-randomization.

1 Introduction

A fundamental problem in robotics is the task planning problem. Given the models of the robot manipulator and the environment in which it operates, the problem is to generate a sequence of actions to accomplish a given task. For assembly, material handling, spot welding, measuring, testing, and inspecting one wants to generate a continuous intersection-free motion of the robot manipulator that connects several given configurations of the end-effector.

In this work, the energy-optimal motion planning problem for planar robot manipulators with two revolute joints is studied. In addition, the end-effector of the robot manipulator is constrained to pass through a set of waypoints, whose sequence is not predefined. We propose a multi-start approach to solve the problem that determine the promising discrete path and evaluate the continuous dynamic trajectory.

The paper is organized as follows. The robot motion planning problem studied in this paper will be stated in Section 2. Our Multi-Start approach is described in Section 3, for which we previously motivate the basis. In Section 4 the results obtained applying our approach to several instances are reported. Finally, some conclusions will be drawn in Section 5.

2 The Robot Motion Planning Problem

The motion planning problem is an optimal control problem of a mechanical system. Each system comprises a dynamic model to take into account [1] and [2]. That is, besides geometrical feasibility, it is also important to ensure dynamical feasibility. Related to optimality, the motion planning must be executed with minimum energy consumption. Furthermore, being a dynamic system, the solution of the problem must provide the optimal scheme of accelerations and velocities during the motion.

In particular, the mechanical system for which we attempt optimal control is a planar robot manipulator with two revolute joints, which we will denote as *RR*. The *RR* robot qualitatively corresponds to the model of the first two links of a SCARA (Selective Compliant Assembly Robot Arm) without taking into account the vertical one, see the left-hand side of Figure 1. And, however this simple robot manipulator can be, it has a very complex nonlinear dynamics and comprises most of the kinematical and dynamical properties of a typical industrial robot. So, the *RR* is composed by two homogeneous links and two actuated joints moving in a horizontal plane $\{x, y\}$, as shown in the right-hand side of Figure 1, where l_i is the length of link i, r_i is the distance between joint i and the mass center of link i, and θ_i is the angular position of link i, for $i = 1, 2$. Finally, the vector $\tau = (\tau_1, \tau_2)$ defines the control inputs of the system, where τ_1 is the torque applied by the actuator at joint 1 and τ_2 is the torque applied by the actuator at joint 2.

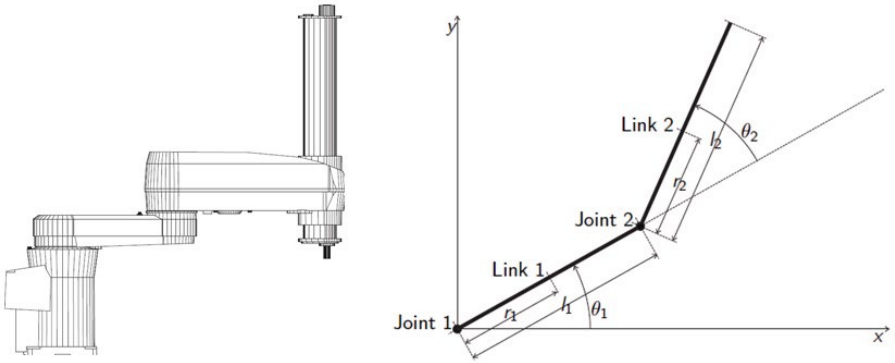

Figure 1. A robot manipulator that moves in a horizontal plane

In this work, the *RR* motion planning problem not only includes its correspondent dynamic model, but also constraints between the initial and final positions. In

particular, the robot manipulator is constrained to pass once through all the given points in the workspace, with no predefined sequence, when moving from an initial position p_I to a final position p_F. The presence of these waypoint constraints adds a combinatorial complexity to this optimal control problem and makes it particularly difficult to solve. In summary, the main features that make this problem especially complex: i) dynamic problem ii) non-linear problem and iii) combinatorial problem.

3 How to solve the problem

We use this section to highlight what it is expected to be a solution of the problem. According to that, we analyze the problem complexities and motivate our approach based on split and conquer strategy. In particular, we split as follows: first, elaborate a Multi-Start algorithm for the combinatorial problem and then, display the nonlinear problem formulation in the IPOPT solver to obtain the dynamic motion of the problem.

3.1 Background and Motivation

In P. Bonami *et al.* [3] they model the problem as a Mix Integer NonLinear Programming. Moreover, the problem is converted into a NonLinear Programming (NLP) problem, when the sequence of waypoints is fixed. So, Bonami *et al.* [3] solve the *RR* optimal control problem for 18 waypoints (see Figure 2) using BONMIN solver [4], which integrates a BB algorithm and the IPOPT solver [5] for NLP. It is important to point out that, since both the order in which the waypoints are visited and the corresponding velocities are not specified, they must be determined. As far as we know, for the latter it is necessary the use of optimization engines such as IPOPT to solve the NLP.

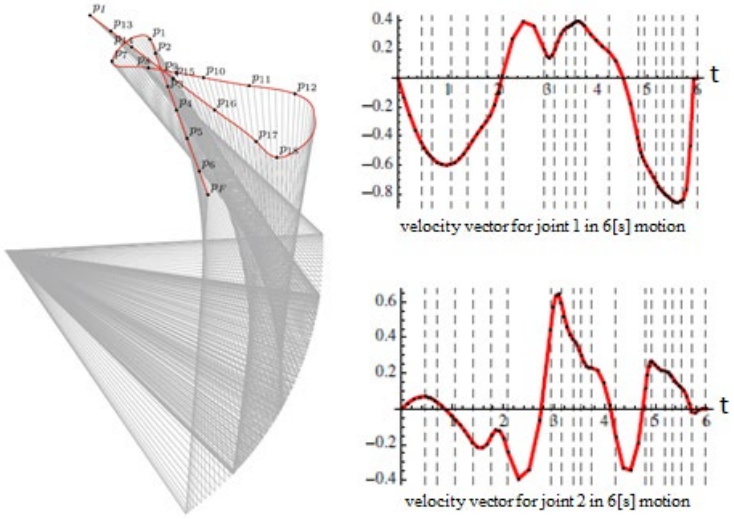

Figure 2. BONMIN pseudo-optimal trajectory and control variables within 18 waypoints.

On the other hand, we observe that the continuous path is strongly affected by the discrete path, i.e. the sequence of points. In particular, we experimentally checked that main energy consumption is due to changes in velocity which occur when there is a change in the direction of the motion. Therefore, the preferable sequence of waypoints for minimizing energy consumption will comprise straight paths within smooth turns.

So, in our approach will take advantage of that property and look for promising discrete paths, unlike the BB algorithm [3] which does not make distinction between them. In other words, our approach is based on finding these straight paths and avoiding sharp turns, taking into account the location of the waypoints. Once the promising discrete path have been found, the corresponding continuous optimal trajectories (i.e. continuous paths plus velocity profile along them) is found by solving the NLP problem. In fact, when the sequence of waypoints is fixed, IPOPT solver finds for the NLP problem good dynamic trajectories in short computational times.

3.1 Multi-Start Approach

In order to construct sequences of waypoints which include the minimum changes of direction as possible, we will look for straight segments and then join the segments and the isolated points. The pseudocode for this algorithm is depicted in Figure 3, and basically, this approach includes the procedures that follow:

1. *Search possible segments in the workspace*. In this procedure we consider each edge from the workspace and compute one by one the acute angle respect to all the other waypoints. In case we obtain 180º the waypoint will be included. We find a segment when there are at least two more waypoints included in the edge.
2. *Use biased randomization*. Once we obtain the list of segments, we sort that list using a skewed probability distribution. A skewed distribution is used here in order to assign higher probabilities of being in the top of the list by segments composed with a larger number of waypoints. In our case, a Geometric distribution with $\alpha = 0.25$ was employed to induce this biased-randomization behavior [6]. So, we always select the segment in the top to be the discrete path solution. Next, we update the list of segments, since the waypoints from the segment selected must be removed from the other segments. Then, sort the list and pick the one in the top again. Step 2 ends when the list of segments is empty.
3. *Join segments and isolated waypoints*. Once we have a "good" selection of segments to be in the discrete path, we compute Euclidean distance for all possible connections, i.e. ends of the segments and isolated waypoints. And we sort the list of possible connections using a biased-randomization of a geometric distribution with $\beta = 0.25$. So, we always join using the connection in the top of the Euclidean distance list. Next, we update the list of connections, since there would be connections that must be removed after we have joined. Then, sort the list and join using the one in the top again. Step 3 ends when all the waypoints are connected in a discrete path.
4. *Repeat step 2 and 3*. After having studied the parameter analysis, the proper stopping criteria for this Multi-Start approach for *RR* problems with 12-18

waypoints, are maximum 10-15 iterations for step 2 and maximum 30 iterations for step 3. It means, for each group of selected lines in step 2, we will try 30 different connections in step 3. So, the Multi-Start approach will create 10-15 x 30 discrete paths, although in many cases we obtain a repeated one from previous iterations.

5. *Sort discrete paths*. We calculate the total acute angle in every distinguished discrete path. The total acute angle is the addition of each acute angle, i.e. acute angle of each 3 consecutive waypoints in the discrete path. And, we sort the discrete path from the lowest to the highest total angle computed.
6. *Solution of the RR.* We call IPOPT through the correspondent .nl file within the NLP problem formulation. Because the discrete path will be fixed according to the list of path, the IPOPT solver provides trajectories quickly. In *RR* problems with 12-18 waypoints, it will be enough checking the first 30 paths from the list.

```
Multi-StartApproach (waypoints, alpha, beta, maxIter2, maxIter3, maxIpopt, nlfile)

% Searching for good discrete paths
edges <- read(waypoints)
segments <- findSegments (edges, waypoints) % step 1
for { 0 to maxIter2 } do
   while { segments list is not empty } do
      randomSelection (segments, alfa) % step 2
      update(segments)
   end while
   distanceList <- calcDist(waypoints isolated & end segments)
   for { 0 to maxIter3 } do
      while { all waypoints not in discrete path} do
         randomSelection(distanceList, beta) % step 3
         update (distanceList)
      end while
      pathList <- save (currentPath) % sorted list of new solutions
   end for % step 4
end for % step 4

% Evaluating the obtained discrete path
calcTotalAngle(pathList)
sortDiscretePaths(pathList) % step 5
energy(bestTrajectory) <- infinite
for { it = 0 to maxIpopt } do
   Trajectory [it] <- solutionIpopt(pathList[it], nlfile) % step 6
   if {energy(Trajectory [it]) < energy(bestTrajectory)} then
      bestTrajectory <- Trajectoty[it]
   end if
end for
return bestTrajectory
```

Figure 3. Pseudocode for the *RR* pseudo-optimal solution, i.e. continuous trajectory

Finally, to mention that in step 1 the allowable acute angle could be variable. In this case we have considered 180º, i.e. completely straight segments, since the waypoints

in the workspace would be somehow aligned. However, in case that the waypoints are not absolutely aligned we just need smoothly decrease the acute angle in order to enable quasi-straight segments.

3.3 Problem Formulation for IPOPT solver

In this section we summarize the problem formulation of the optimal control problem and the fundamental characteristics studied in [3]. First of all, we present the continuous optimal control problem formulation:

$$\text{Min } J\,[x(t), u(t), s] := E\,[x(t_F), s] + \int L\,[x(t), u(t), s]\,dt \qquad (1)$$

subject to:

$$\dot{x}(t) = f\,[x(t), u(t), s], \qquad t \in [t_I, t_F] \qquad (2)$$

$$0 = g\,[x(t), u(t), s], \qquad t \in [t_I, t_F] \qquad (3)$$

$$0 \leq c\,[x(t), u(t), s], \qquad t \in [t_I, t_F] \qquad (4)$$

$$r^{ineq}\,[x(t1), x(t2), \ldots, x(t_{nrineq}), s] \leq 0 \qquad (5)$$

$$r^{eq}\,[x(t1), x(t2), \ldots, x(tnreq), s] = 0 \qquad (6)$$

$$x(t_I) = x_I \qquad (7)$$

$$\psi\,[x(t_F)] = 0 \qquad (8)$$

The objective functional in (1) is given in Bolza form and it is expressed as the sum of the Mayer term, which is assumed to be twice differentiable, and the Lagrange term. Variable $t \in [t_I, t_F]$ represents time, where t_I and t_F are the initial and final time, respectively. $x(t)$ represents the state variables within both, differential and algebraic variables and $u(t)$ represents the control functions, also referred to as control inputs, which are assumed to be measurable. The vector s contains all the time-independent variables of the problem. Equation (2) and (3) represents a Differential Algebraic Equations (DAE) system. The function f is assumed to be piecewise Lipschitz continuous to ensure existence and uniqueness of a solution. The system must satisfy the algebraic path constraints c in (4) and the interior point inequality and equality constraints r^{ineq} and r^{eq} in constraints (5) and (6), respectively, which are assumed to be twice differentiable. Finally, x_I in (7) represents the vector of initial conditions given at the initial time t_I and the function ψ in (8) provides the terminal conditions at the final time t_F, which is assumed to be twice differentiable.

In addition, we can introduce integer variables to obtain a multi-phase problem, and therefore, the so-called mixed integer optimal control problem appears. Moreover, we can convert the mixed-integer optimal control problem into a mixed integer nonlinear programming problem including these transformations: i) making unknown passage times through the waypoints part of the state, ii) introducing binary variables to enforce the constraint of passing once through each waypoint, and iii) applying a fifth-degree Gauss-Lobatto direct collocation method to tackle the dynamic constraints. High degree interpolation polynomials allow the number of variables of the problem to be reduced for a given numerical precision. Finally, the problem will became a NLP problem with non-convex feasible region as far as we fix

the sequence of waypoints. And the solution for this NLP problem is achieved in IPOPT solver in a short time. Further explanation about the optimal control problem transformations are in [3].

4 Numerical experiments

In this section, the results of several numerical experiments where the robot is constraint to pass through the waypoints listed in Table 1 for the testbed lines (LIN) and listed in Table 2 for the testbed lattice (LAT).

p_1 = (0.455718, 0.660622)	p_2 = (0.472266, 0.616427)	p_3 = (0.510878, 0.513305)
p_4 = (0.538458, 0.439647)	p_5 = (0.571554, 0.351256)	p_6 = (0.610167, 0.248135)
p_7 = (0.335096, 0.591699)	p_8 = (0.450359, 0.571340)	p_9 = (0.536806, 0.556071)
p_{10} = (0.623253, 0.540801)	p_{11} = (0.767332, 0.515353)	p_{12} = (0.911410, 0.489904)
p_{13} = (0.331795, 0.685981)	p_{14} = (0.397159, 0.636696)	p_{15} = (0.527889, 0.538125)
p_{16} = (0.658618, 0.439554)	p_{17} = (0.789347, 0.340984)	p_{18} = (0.854711, 0.291699)

Table 1. Coordinates of the waypoints used in the experiments LIN

p_1 = (-0.181751, 0.581213)	p_2 = (-0.111041, 0.510503)	p_3 = (-0.0403301, 0.439792)
p_4 = (0.0303806, 0.369081)	p_5 = (0.101091, 0.29837)	p_6 = (0.171802, 0.22766)
p_7 = (-0.111041, 0.651924)	p_8 = (-0.0403301, 0.581213)	p_9 = (0.0303806, 0.510503)
p_{10} = (0.101091, 0.439792)	p_{11} = (0.171802, 0.369081)	p_{12} = (0.242513, 0.29837)
p_{13} = (-0.04033, 0.722635)	p_{14} = (0.0303806, 0.651924)	p_{15} = (0.101091, 0.581213)
p_{16} = (0.171802, 0.510503)	p_{17} = (0.242513, 0.439792)	p_{18} = (0.313223, 0.369081)

Table 2. Coordinates of the waypoints used in the experiments LAT

In Table 3 we report the results obtained while using the biased randomization in the *RR* problem. It has been implemented in a C++ code which generates a set of good discrete path where those with minimum total angle are evaluated in IPOPT optimization engine. For each instance we indicate the correspondent testbed and the number of waypoints considered. In all the experiments we use the initial time $t_I = 0$ [s] and the final times $t_F = 4$ [s] and $t_F = 6$ [s] for instances of 12 and 18 waypoints, respectively. Also, instances with cases A, B and C differ in the location of the final point p_F which will be specified in each instance case.

Finally, Figure 4 shows the lines (LIN) and lattice (LAT) configurations. Also, depicts the continuous path of minimum energy consumption in couple of instances, according to our Multi-Start approach.

Instance	initial point and final point sequence of waypoints from Multi-Start
LIN-12A	p_I = (1.0843, 0.459365) p_F = (0.637747, 0.174476) (p_I, p_{12}, p_{11}, p_{10}, p_9, p_8, p_7, p_1, p_2, p_3, p_4, p_5, p_6, p_F)

LIN-12B	$p_I = (1.0843, 0.459365)$ $p_F = (0.2027, 0.6151)$ $(p_I, p_{12}, p_{11}, p_1, p_2, p_3, p_4, p_5, p_6, p_{10}, p_9, p_8, p_7, p_F)$
LIN-18A	$p_I = (0.2664, 0.7353)$ $p_F = (0.637747, 0.174476)$ $(p_I, p_{13}, p_{14}, p_{15}, p_{16}, p_{17}, p_{18}, p_{12}, p_{11}, p_{10}, p_9, p_8, p_7, p_1, p_2, p_3, p_4, p_5, p_6, p_F)$
LIN-18B	$p_I = (0.2664, 0.7353)$ $p_F = (0.933149, 0.232556)$ $(p_I, p_{13}, p_7, p_9, p_{10}, p_{11}, p_{12}, p_6, p_5, p_4, p_3, p_2, p_1, p_{14}, p_8, p_{15}, p_{16}, p_{17}, p_{18}, p_F)$
LIN-18C	$p_I = (0.2664, 0.7353)$ $p_F = (1.0843, 0.459365)$ $(p_I, p_{13}, p_{14}, p_{15}, p_{16}, p_{17}, p_{18}, p_6, p_5, p_4, p_3, p_2, p_1, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}, p_F)$
LAT-12	$p_I = (1.0843, 0.459365)$ $p_F = (0.637747, 0.174476)$ $(p_I, p_1, p_8, p_9, p_{10}, p_{11}, p_{12}, p_6, p_5, p_4, p_3, p_2, p_7, p_F)$
LAT-18	$p_I = (1.0843, 0.459365)$ $p_F = (0.637747, 0.174476)$ $(p_I, p_1, p_8, p_9, p_{10}, p_{11}, p_{17}, p_{16}, p_{15}, p_{14}, p_{13}, p_7, p_2, p_3, p_4, p_5, p_6, p_{12}, p_{18}, p_F)$

Table 3. Multi-Start approach pseudo-optimal discrete paths

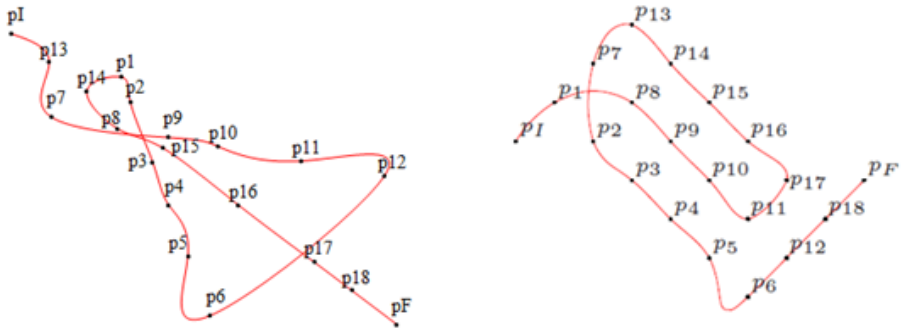

Figure 4. Continuous path for instances LIN-18B and LAT-18 using biased randomization

In order to compare the previous work with the BONMIN solver and our Multi-Start approach we applied both methods in each of the instances in Table 3. Notice that any of these methodologies is an exact method, the first one is based on BB algorithm and ours is meta-heuristics approach, so the optimal solution not guarantee in any case.

Regarding to optimality, in both procedures we have obtained the same discrete path, except for LIN-18B and LAT-12A. And therefore, the same continuous trajectory and energy consumption returned from IPOPT, except for LIN-18B and LAT-12A. In LIN-18B the discrete path obtained in BONMIN is (p_I, p_{13}, p_1, p_3, p_4, p_5, p_6, p_{12}, p_{11}, p_{10}, p_9, p_8, p_7, p_{14}, p_2, p_{15}, p_{16}, p_{17}, p_{18}, p_F) were the energy consumption is

reduced from our solution 2,29%. However, in LAT-12A the discrete path obtained in BONMIN is (p_I, p_1, p_2, p_7, p_3, p_4, p_5, p_6, p_{12}, p_{11}, p_{10}, p_9, p_8, p_F) were the energy consumption has increased from our solution more than 5%.

Regarding to computational times, our Multi-Start approach provides us the solution in few minutes. In a similar CPU, the BONMIN solver is also able to solve lines instances (LIN) in few minutes, but dozen of hours for the lattice configuration. However, the latter is the realistic configuration that often occurs in the industrial robots.

5 Conclusions

This paper we study the motion planning problem where to face high complexities. In one hand the combinatorial problem, on the other hand the dynamic model. For the latter, most of the methods interpolation to determine velocity profile along the path, however the resulting trajectory can be dynamically unfeasible due to physical limitations of the actuators. In this work we proposed a Multi-Start approach which provides not only the discrete path but also ensure the dynamical feasibility of the trajectory. Therefore, we apply biased randomization to obtain a set of good discrete paths and the NLP problem to be displayed in IPOPT solver to obtain the dynamicity of the trajectory. And, since we evaluate continuous trajectories only if the discrete path is promising, the numerical experiments in terms of computational times are much lower than those in [3] where there is no discrimination in the use of IPOPT solver.

Acknowledgment

This work has been partially supported by the Spanish Ministry of Economy and Competitiveness (TRA2013-48180-C3-P and TRA2015-71883-REDT), FEDER, and the Catalan Government (2014-CTP-00001).

References

1. La Valle, S. Planning Algorithms. Cambridge University Press, (2006).
2. Siciliano, B. and Khatib, O. editors. Handbook of Robotics. Springer, (2008).
3. Bonami, P., Olivares, A. and Staffetti, E. Energy-optimal multi-goal motion planning for planar robot manipulators. Journal of Optimization Theory and Applications, 163(1), 80--104, (2014).
4. Bonami, P., Biegler, L.T., Conn, A.R., Cornuéjols, G., Grossmann, I.E., Laird, C.D., Lee, J., Lodi, A., Margot, F., Sawaya, N. and Wächter, A. An algorithmic framework for convex mixed integer nonlinear programs. Discrete Optimization, 5(2), 186--204, (2008).
5. Pirnay, H., López-Negrete, R. and Biegler, L.T. sIPOPT Reference Manual. Carnegie Mellon University, (2011). https://projects.coin-or.org/Ipopt
6. Juan, A., Faulin J., Ruiz R., Barrios B. and Caballé S. The SR-GCWS hybrid algorithm for solving the capacitated vehicle routing. Applied Soft Computing 10, 215—224, (2010).