CONTROLO 2006

7TH PORTUGUESE CONFERENCE ON AUTOMATIC CONTROL

LISBON • INSTITUTO SUPERIOR TÉCNICO • CONGRESS CENTER

11 - 13 SEPTEMBER 2006

ORGANIZED BY

SUPPORT BY

CONTROLO 2006
7th Portuguese Conference on Automatic Control
Instituto Superior Técnico, Lisboa, Portugal
September 11-13, 2006

PARTICLE SWARM OPTIMIZATION *VERSUS* GENETIC ALGORITHM IN MANIPULATOR TRAJECTORY PLANNING

E. J. Solteiro Pires * **P. B. Moura Oliveira** *
J. A. Tenreiro Machado ** **J. Boaventura Cunha** *

* *Dep. Eng. Electrotécnica, UTAD University, 5000-911 Vila Real, Portugal*
** *Dep. Eng. Electrotécnica, Inst. Sup. de Eng. do Porto, R. Dr Ant. Bernadino Almeida, 4200-072 Porto, Portugal*

Abstract: The aim of this study is the reduction of the computational burden associated with the evolutionary optimization of manipulator trajectory planning. This paper proposes the use of a particle swarm optimization algorithm to generate trajectories for robotic planar manipulators, based on direct kinematics. The design objective is to minimize the ripple in the trajectory time evolution. Several redundant and hyper-redundant manipulators are considered. The particle swarm optimization algorithm is compared with genetic algorithm in solving the manipulator trajectory planning problem. Preliminary simulation results are presented.

Keywords: Particle Swarm Optimization, Genetic Algorithms, Robotics, Trajectory Planning, Optimization.

1. INTRODUCTION

Evolutionary inspired algorithms, particulary genetic algorithms (GAs) have proved to be a good optimization tool in robotic applications including manipulator trajectory planning (Davidor (1991); Rana and Zalzala (1996); Kubota *et al.* (1996); Chen and Zalzala (1997)). This problem can be tackled by using the differential inverse kinematics for generating the manipulator trajectory (Chen and Zalzala (1997); Davidor (1991)). However, the former approach must take into account the problem of kinematics singularities which can be a serious drawback. To avoid this problem other techniques were proposed (Rana and Zalzala (1996); Kubota *et al.* (1996)) which are based on the use of direct kinematics.

The type of manipulator trajectory planning problem addressed in this study (described in detail in section 2) was solved successfully by using GAs considering both a single objective (Solteiro Pires *et al.* (2001)) and multi-objective (Solteiro Pires *et al.* (2004)) formulation. The problem associated with the use of evolutionary based algorithms to solve manipulator trajectory planning problems is that they still are very time consuming. The reduction of the computational load involved with the use of GAs can be crucial within practical industrial applications. Thus, the study of alternative evolutionary inspired algorithms which are conceptually simpler than GAs and less demanding in terms of computational load is of great interest. Among the most successfully natural inspired algorithms proposed recently is the particle swarm optimization (PSO) (Kennedy and Eberhart (1995)). Indeed, some robotics applications wich incorporates the use of PSO have been proposed. Tan *et al.* (Tang *et al.* (2005)) presents a path planning approach based on a neural network model and PSO in order to evolve smooth motion paths quickly. The workspace was modeled using actual point coordinates (x, y). These points were fixed at equally spaced x-coordinates and

the y-coordinates served as the control values to be optimized. PSO was used to evolve these values hoping to find a better path. The (x, y) coordinates of the path points were then fed into a neuronal network.

Wu *et al.* (Wu *et al.* (2004)) used a PSO to optimize a trajectory for a dual-arm space robot. The motion-planning problem was treated as a constrained continuous optimization problem. The path was modeled using Bspline curves and the control points are the parameters to be optimized. The function to be minimized is the vibrations caused by the robot movement. Constraints were applied to limit the values of the joint angles and the velocities and accelerations of the robot joints.

This paper reports the use of a standard PSO algorithm to solve the manipulator trajectory planning problem. Following this introduction the proposed trajectory planning scheme is described in section 2, simulation results are presented in section 3. Finally, in section 4 some conclusion are drawn.

2. THE TRAJECTORY PLANNING SCHEME

The proposed trajectory planning strategy using a stochastic optimization technique is explained in this section. The optimized trajectory is required to have a reduced ripple in the space/time evolution. Two, three and four link manipulators are considered, which are required to move between two points in the workspace. Two stochastic algorithms: the particle swarm optimization (PSO) and the genetic algorithm (GA) are used as optimization tools. Direct kinematics are deployed to avoid singularity problems.

2.1 Trajectory Representation

The manipulator can move between two points of the workspace. Therefore, the initial and final configurations are given by the inverse kinematic equations. The path is encoded directly, using real codification, as strings in the joint space to be used by the algorithm as:

$$\{(q_{11}, \ldots, q_{l1}), \ldots, (q_{1n}, \ldots, q_{ln})\} \tag{1}$$

The ith joint variable for a robot intermediate jth position is q_{ij}, the vector is constituted by l configurations and each configuration has n joint arm values. The joint variables q_{ij} are initialized in the range]-180^o,+180^o]. It is important to note that the initial and final configurations have not been encoded into the string because this configuration remains unchanged throughout the trajectory search. The time between two consecutive configurations is $\Delta t = 0.1$s.

In order to apply the PSO to solve this manipulator optimization problem $n = 8$ parameters (1) are encoded as a swarm particle (or GA population element) represented as a vector of real values.

2.2 Optimization criteria

Four indices are used to qualify the evolving robotic manipulators trajectories. All indices are translated into penalty functions to be minimized. Each index is computed individually and is integrated in the fitness function evaluation.

The fitness function f adopted for evaluating the candidate trajectories is defined as:

$$f = \beta_1 f_q + \beta_2 f_{\dot{q}} + \beta_3 f_p + \beta_4 f_{\dot{p}} \tag{2}$$

where the indices f_q, $f_{\dot{q}}$, f_p, $f_{\dot{p}}$ are defined in the sequel. The optimization goal consists in finding a set of design parameters that minimize f according to the priorities given by the weighting factors β_i $(i = 1, .., 4)$.

The f_{q} function is used to minimize the manipulator traveling distance yielding the criteria:

$$f_q = \sum_{j=1}^{n} \sum_{i=1}^{l} \dot{q}_{ij}^2 \tag{3}$$

This equation is used to optimize the traveling distance because, if the curve length is minimized, the ripple in the space trajectory is indirectly reduced. For a function $y = g(x)$ the distance curve length is $\int[1 + (dg/dt)^2]\, dx$ and, consequently, to minimize the distance curve length it is adopted the simplified expression $\int(dg/dt)^2\, dx$. The fitness function maintains the quadratic terms so that the robot configurations are uniformly distributed between the initial and final configurations.

The $f_{\dot{q}}$ function are used to minimize the ripple in the time evolution of the robot trajectory through the criteria:

$$f_{\dot{q}} = \sum_{j=1}^{n} \sum_{i=1}^{l} \ddot{q}_{ij}^2 \tag{4}$$

The cartesian distances are introduced in the fitness function f to minimize the total trajectory length, from the initial point up to the final point. This criterion is defined as:

$$f_p = \sum_{j=2}^{n} d\left(p_j,\ p_{j-1}\right)^2 \tag{5}$$

where p_j is the robot j intermediate arm cartesian position and $d(\cdot,\cdot)$ is a function that gives the distance between the two arguments.

The $f_{\dot{p}}$ function in the fitness functions is responsible for reducing the ripple in time evolution of the arm velocities. This index is formulated as:

$$f_{\dot{p}} = \sum_{j=3}^{n} \left| d\left(p_j,\ p_{j-1}\right) - d\left(p_{j-1},\ p_{j-2}\right) \right|^2 \tag{6}$$

2.3 The Particle Swarm Optimization

The particle swarm optimization algorithm was proposed originally by Kennedy and Eberhart (Kennedy and Eberhart (1995)). This optimization technique is inspired in the way swarms (flocks of birds, schools of fishes, herds, *etc.*) elements move in a synchronized way as a defensive tactic. An analogy is established between a particle and an element of swarm. The particle movement is characterized by two vectors representing its current position x and velocity v.

Since 1985 many techniques have been proposed to refine and/or complement the original PSO algorithm, namely regarding tuning parameters (**?**) and by considering hybridization with other evolutionary techniques (**?**).

In this study a standard elementary PSO is considered (see algorithm 1). The basic algorithm begins by initializing the swarm randomly in the search space. As it can be seen in algorithm 1, each particle position is changed in every iteration by adding a new velocity. This velocity is evaluated by summing an increment to the old velocity value. The increment is a function of two components representing the cognitive knowledge and the social knowledge.

The cognitive knowledge of each particle is incorporated by evaluated the difference between the current position x and its best position so far b. The social knowledge of each particle is incorporated by evaluated the difference between its current position x and the best swarm global position achieved so far g. The cognitive and social knowledge factors are multiplied by a randomly generated constant ϕ_1 and ϕ_2, respectively. The velocity of particles are restricted in order to keep velocities from exploding.

Initialize Swarm;
repeat
  forall *particles* **do**
    calculate fitness f
  end
  forall *particles* **do**
    $v_{t+1} = v_t + \phi_1(b - x) + \phi_2(g - x)$
    $v_{t+1} \in [-v_{\text{min}}, +v_{\text{min}}]$
    $x_{t+1} = x_t + v_{t+1}$
  end
until *stopping criteria* ;
Algorithm 1: Particle swarm optimization

2.4 Genetic Algorithm

Standard genetic algorithm (GA) is another optimization technique. Initially, the GA initializes randomly the population P(t). Each population element is represented computationally by a data structure which incorporates problem dependent parameters. In one generation these data structures are crossed between them and are subjected to a mutation process in order to produce better offsprings.

$t = 0$
random ($P(t)$)
fitness($P(t)$)
repeat
  select $P(t+1)$ from $P(t)$
  crossover($P(t+1)$)
  mutate($P(t+1)$)
  fitness($P(t+1)$)
  $t = t + 1$
until *conclusion condition* ;
Algorithm 2: Genetic Algorithm

The selection operator used implemented uses a linear ranking scheme. A binary crossover simulated operator (**?**) and a uniform base mutation are used.

3. SIMULATION RESULTS

The experiments consist on moving a robotic arm from the starting point $A \equiv \{-0.5, 1.4\}$ up to the final point $B \equiv \{1.2, -0.3\}$. The simulations results were achieved by using the following settings, with $n = 8$ configurations, $T = 8000$ iterations. The swarm/population size is $pop_{\text{size}} = 100$. Parameters ϕ_1 and ϕ_2 are uniformly generated in the interval $[-0.8, 0.8]$. The weights values used in the aggregated function (2) are $\beta_i = \{1, 1, 0.5, 0.5\}$ for $i = \{1, 2, 3, 4\}$.

This section presents the results of several simulations for two optimization types, that is, using the PSO and the GA. Figures 1 to 3 shows the two-link manipulator results: the successive configurations, the angular joint positions and the velocity joints, respectively.

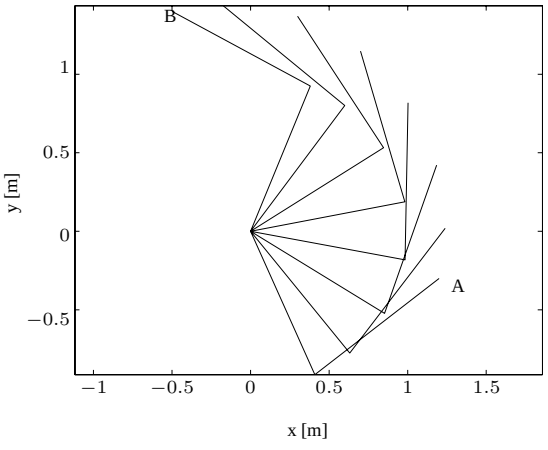

Fig. 1. 2-dof robotic manipulator successive configurations

The results obtained with PSO in terms of minimal joint/cartesian distance and joint/cartesian ripple, for the weighings settings used, are of good quality and

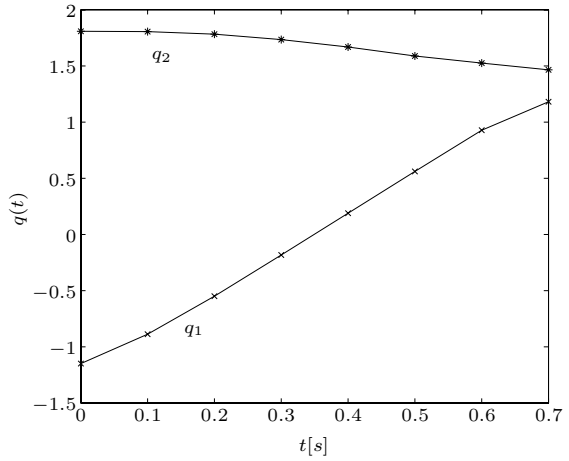

Fig. 2. 2-dof robotic manipulator successive configurations

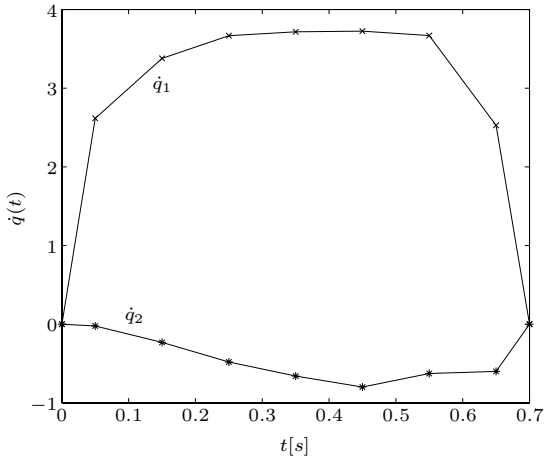

Fig. 3. Angular velocity *versus* time of 2-dof robotic manipulator

similar to the ones achieved with a GA (Solteiro Pires *et al.* (2004)).

Figure 4 shows the successive configurations obtained for the three-link manipulator.

Tables 1 and 2 present the results obtained with the PSO and GA, respectively. In these tables the *fitness* represents the best result achieved for the aggregated cost function (2), and the term *time* refers to the duration of the simulation. While the best fitness obtained is identical with both algorithms the time elapsed in the runs is smaller in the PSO simulations. The simulation time improvement achieved with the PSO is partly justified because this algorithm is simpler than the GA.

Figure 5 shows the best fitness values *versus* number of generations using PSO and GA for all the manip-

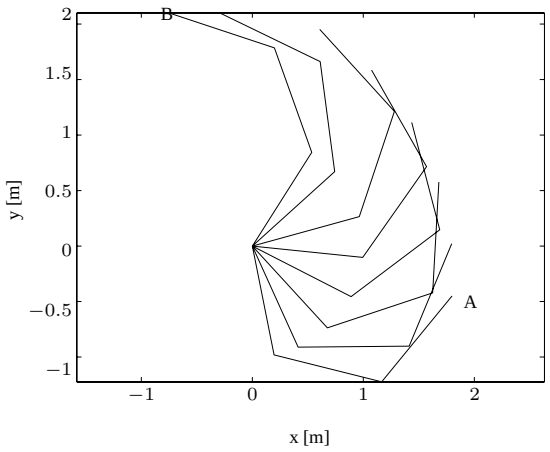

Fig. 4. 3-dof robotic manipulator successive configurations

Table 1. Fitness of the best particle found in the algorithm and the time required by the algorithm for the 2, 3 and 4-dof robot.

PSO	2-dof	3-dof	4-dof
fitness	1.07	1.42	1.95
time [s]	12	16	21

Table 2. Fitness of the best individual found in the genetic algorithm and the time required for the 2, 3 and 4-dof robot.

AG	2-dof	3-dof	4-dof
fitness	1.07	1.42	1.94
time [s]	20	25	31

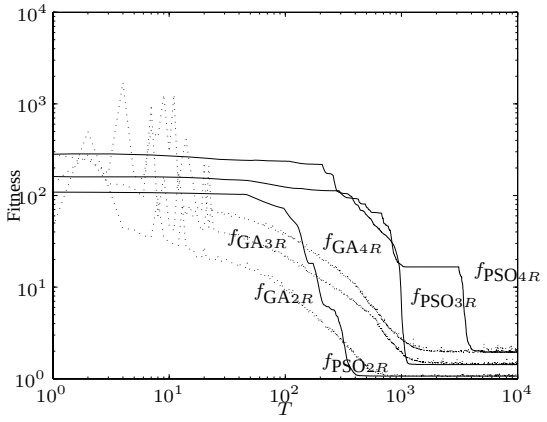

Fig. 5. Log fitness *versus* log generations

ulators studied. As can be seen for two out of three simulations the PSO is faster to converge than the GA.

4. CONCLUSIONS

This paper proposed a particle swarm algorithm to generate trajectories for robotic planar manipulators, based on direct kinematics. The objective is to minimize distance and the ripple in the trajectory time

evolution. Some results was presented for several redundant and hyper-redundant manipulators. The particle swarm optimization algorithm is compared with genetic algorithm in solving the manipulator trajectory planning problem. The preliminary simulation results obtained have a good performance and indicate that the PSO is faster than the GA with a lower computational load.

5. FUTURE WORK

Further work is undergoing to re-enforce these preliminary results achieved by PSO which will allow to conclude about its effectiveness in manipulator trajectory optimization both for a single and multiple criteria optimizations.

REFERENCES

Chen, Mingwu and Ali M. S. Zalzala (1997). A genetic approach to motion planning of redundant mobile manipulator systems considering safety and configuration. *Journal Robotic Systems* **14**(7), 529–544.

Davidor, Yaval (1991). *Genetic Algorithms and Robotics, a Heuristic Strategy for Optimization*. number 1 In: *Series in Robotics and Automated Systems*. World Scientific Publishing Co. Pte Ltd.

Kennedy, James and Russell C. Eberhart (1995). Particle swarm optimization. In: *Proceedings of the 1995 IEEE International Conference on Neural Networks*. Vol. 4. Perth, Australia, IEEE Service Center, Piscataway, NJ. pp. 1942–1948.

Kubota, Naoyuki, Toshio Fukuda and Koji Shimojima (1996). Trajectory planning of cellular manipulator system using virus-evolutionary genetic algorithm. *Robotics and Autonomous systems* **19**, 85–94.

Rana, A. and A. Zalzala (1996). An evolutionary planner for near time-optimal collision-free motion of multi-arm robotic manipulators. In: *UKACC International Conference on Control*. Vol. 1. pp. 29–35.

Solteiro Pires, E. J., J. A. Tenreiro Machado and P. B. de Moura Oliveira (2001). An evolutionary approach to robot structure and trajectory optimization. ICAR'01-10th International Conference on Advanced Robotics. Budapest, Hungary. pp. 333–338.

Solteiro Pires, E. J., P. B. de Moura Oliveira and J. A. Tenreiro Machado (2004). Multi-objective genetic manipulator trajectory planner. In: *Applications of Evolutionary Computing, EvoWorkshops2004: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoMUSART, EvoSTOC*.

Tang, J., J. Zhu and Z. Sun (2005). A novel path panning approach based on appart and particle swarm optimization. In: *Proceedings of the 2nd International Symposium on Neural Networks* (LNCS, Ed.). Vol. 3498.

Wu, H., F. Sun, Z. Sun and L. Wu (2004). Optimzal trajectory planning of a flexible dual-arm space robot with vibration reduction. *Journal of Intelligent Robotic Systems* **40**, 147–163.