The International Journal of Advanced Manufacturing Technology manuscript No.
(will be inserted by the editor)

Statistical evaluation of an evolutionary algorithm for minimum time trajectory planning problem for industrial robots

Fares J. Abu-Dakka[1] · Iyad F. Assad[2] · Rasha M. Alkhdour[1] · Mohamed Abderahim[1]

Received: date / Accepted: date

Abstract This paper presents, evaluates and validates a genetic algorithm procedure with parallel-populations for the obtaining of minimum time trajectories for robot manipulators. The aim of the algorithm is to construct smooth joint trajectories for robot manipulators using cubic polynomial functions, where the sequence of the robot configurations is already given. Three different types of constraints are considered in this work: (1) Kinematics: these include the limits of joint velocities, accelerations, and jerk. (2) Dynamic: which include limits of torque, power and energy. (3) Payload constraints. A complete statistical analysis using ANOVA test is introduced in order to evaluate the efficiency of the proposed algorithm. In addition, a comparison analysis between the results of the proposed algorithm and other different techniques found in the literature is described in the experimental section of this paper.

Keywords Industrial robots - Minimum-time trajectory planning - Obstacle avoidance

1 Introduction

Over the last few decades, industrial manipulators have become a commonly used element in automated production lines. They are highly nonlinear multivariable coupled systems, which are often subjected to complex nonlinear constraints. Due to the mentioned complexity, indirect methods (decoupled approaches) for trajectory planning are frequently used. Indirect methods solve the planning problem in two steps: planning the path and then adjusting the trajectory. The path planning step is about finding a collision-free sequence of configurations between an initial and a final configurations of a manipulator, taking into account some aspects such as geometric and kinematic constraints. In the subsequent step, the trajectory is adjusted by optimizing the time along the given path. In the latter step, the optimization problem considers the manipulator dynamics and its actuators' constraints. This type of planning is done offline.

Numerous techniques for optimal offline trajectory planning have been developed in the last decades by different researchers in the field. These techniques can be classified according to the cost function to be minimized, which is often related to the execution time, energy consumption, or jerk. Thus, why offline planning? Many industrial processes are repetitive, e.g., welding, inspection, assembling, painting and even moving objects, etc. which justify the offline trajectory planning of an industrial manipulator. Moreover, the productivity in industry is directly proportional to the speed of operation [34]. In order to maximize the productivity of a robotized operation, it is necessary to maximize the operation speed which implies a minimization of the robot's travelling time. In general, offline planning provides a globally optimized trajectory, smoothness by anticipating sharp corners, and a second-order continuity along the whole trajectory [20]. In addition, if the manipulator workspace and the objective task are completely known, offline planning is more desirable to ensure the minimization of the execution time. Impor-

The authors thank the Ministerio de Economía y Competitividad for financing this work under project RTC-2014-3070-5.

[1]Department of Systems Engineering and Automation of the Carlos III University of Madrid, Av. de la Universidad, 30, 28911 Leganés, Spain.
Tel.: +34-916245925, Fax: +34-916249430
E-mail: fabudakk@ing.uc3m.es (Corresponding Author) ·
[2]Computer Center, Birzeit University, Ramallah, Palestine

tant research work has been done with the objective of achieving minimum time trajectories [37,38,39,40,41].

In order to improve the tracking accuracy and permit the system to reach higher speeds during the execution of tasks, it is important to generate trajectories with a bounded amount of jerk. This definitely entails reducing the robot resonant frequencies excitation and, ultimately reduces the mechanical wear of the whole system [9,15,31,33]. In the literature, the minimum jerk in point-to-point ballistic motion was used as a control variable [54]. In this case, the duration of the motion and initial and final positions are given, while the jerk is the control signal.

In general, every procedure found in the literature that deals with trajectory planning problem has focused on a specific objective function or some parameters for the optimization problem. The most significant among them are (1) minimum execution time, (2) minimum energy, and (3) minimum jerk.

1.1 Minimum time trajectory planning

Minimum time trajectory planning algorithms were the first techniques presented in the literature. The reason for that was their direct relation to increasing the productivity of industrial automation with robots. However, considering the nonlinearity of the systems and joint coupling in robot dynamics, exists the possibility to obtain approximate solutions [21,22]. Other authors in this scope developed interesting techniques in the position-velocity phase plane [6,46]. Alternatively, dynamic programming techniques were employed to solve this problem [4]. However, these algorithms suffer a discontinuity in the torque and acceleration profiles, and this is due to the assumption that robot links were perfectly rigid and neglecting of the actuator dynamics. Later, the authors in [8] overcame these kinds of drawbacks by introducing limits on the actuator jerks. Although, their technique could not be exactly time optimal, but the generated trajectories could be employed in more advanced control strategies. After this short review in such a problem, it can be concluded that to overcome the discontinuity in acceleration of the generated trajectories it is necessary to use smooth functions to interpolate the trajectories such as spline functions. As well as the aim of this paper is to solve minimum time trajectory planning problem using cubic splines, a detailed discussion about splines functions and their use in trajectory generation is presented in Section 1.4.

1.2 Minimum energy trajectory planning

In numerous robotics applications, like those in space or submarine exploration, it is more beneficial to minimize the energy consumption than minimizing the execution time. In such applications, the robotic systems have a limited energy source, which is worth optimizing its use. In scientific literature, reader can find numerous publications in this field. Cubic B-splines functions have been employed to generate trajectories with minimum energy consumptions [45]. However, their algorithm could not achieve smooth accelerations and joint torques. Fourier series expansion is used also with genetic algorithms to search for minimum energy trajectories [19]. Other scientists used fifth order B-splines for minimum energy trajectory generation [16]. Point-to-point trajectory based on minimum absolute input energy is proposed in [13] for an LCD glass-handing robot. Lagrange interpolation method was used too to perform trajectory planning for energy minimization of industrial manipulators [29].

1.3 Minimum jerk trajectory planning

Jerk is the third derivative of the displacement and indicates how rapidly the actuators change their forces. As mentioned in [24], decreasing the jerk leads to decreasing the joint position errors. Moreover, low jerk in joint trajectories is desirable to limit the manipulator vibration which in turn extends the manipulator life-span. However, minimum jerk techniques depend only on the kinematics of the task. In this paper, instead of minimizing the jerk, we have constrained it to some limits given in the literature. In Section 1.4, some approaches to deal with minimum jerk algorithms are discussed.

1.4 State of the art

Earlier trajectory planning models employed nonlinear programming approaches to generate the trajectory, in either gripper [28] or joint [25] space. However, these models do not consider the manipulator dynamics. In this paper, the construction/generation of joint trajectories for industrial manipulators is based on cubic spline functions. Algebraic splines are widely used in path planning, e.g., cubic splines [25] (the focus in this paper), quartic splines [49], quintic splines [16], trigonometric splines [47], and synchronized trigonometric S-curves [30]. In literature, readers can find different types of minimum time joint trajectory planning based on cubic splines. They vary in:

- Types of constraints (kinematics or dynamics),

– The algorithm used in the optimization,
– The optimization problem that can be extended to be more general.

The first case is the most important one which allows making a difference between kinematic and dynamic trajectory planning. Applying kinematic constraints only results a simplified computational model that gives rise to an under capacity utilization of the manipulator, but still produces good solutions. On the other hand, considering dynamic constraints provides better solution because the problem is more defined. However, their computational model is more complex since they deal with some issues such as dynamic identification [14].

Lin et al. work [25] was among the firsts to address the computation of optimal joint trajectories through the interpolation of a sequence of nodes in the joint space. The nodes that form the sequence are obtained from a set of discrete positions of the end-effector of the manipulator. Four years later, Thompson and Patel proposed a quartic B-splines formulation for joint trajectories planning for industrial robots [49]. However, their algorithm could not achieve better execution time than Lin et al.'s algorithm for the same case study. Lin et al.'s algorithm was employed by Wang and Horng [53], where the trajectories are expressed as cubic B-splines. However, their procedure did not produce better execution time than the Lin et al.'s one, but it is faster in terms of computational time cost. The authors in [20] developed an improved version of the optimization algorithm proposed in [25] to include dynamic constraints in the calculations, while others used the interval analysis to solve a minimum time trajectory problem subject to kinematic constraints [32]. On the other hand, Tse and Wang [50] resolved the Lin et al.'s algorithm [25] using genetic algorithms. A detailed comparison of results, between their algorithm and the proposed one in this paper, is shown in Section 4.1. In [10], hybrid optimization procedure is presented to calculate time optimal path-constrained subject to kinematic constraints. However, the resulting trajectories were optimal in terms of time but not smoothness. Gasparetto and Zanotto [16] proposed a minimum jerk trajectory planning technique, which assumes that the geometric path is formed by a sequence of via points in the operating space of the manipulator. However, their work considered only kinematic constrains. Their algorithm uses fifth order B-splines for the trajectory generation. One year later, the same authors resolved the same objective function using a cubic polynomial for trajectory construction [17]. Their results are compared with the proposed algorithm's results in Section 4.4. Readers are advised to refer to [11,23,48,51] for further reading about algorithms that solve minimum time trajectories subject to kinematic constraints. Recently, a minimum jerk joint trajectory using particle swarm optimization (PSO) was proposed in [26]. To enhance the computational performance, the authors integrated the K-means clustering with PSO algorithm to select a possible particle with the least fitness value.

Robot dynamics were considered in the work presented by Chettibi et al. [7], where they developed a sequential quadratic programming (SQP) method for optimal motion planning for a PUMA560 robot manipulator. Among later methods considering dynamics, the authors in [36,35] used harmonic functions to interpolate a sequence of configurations in order to construct manipulator trajectories.

The methods used in the literatures such as sequential unconstrained minimization technique (SUMT) [37,38,39,40,41], SQP [7,17], interval analysis [33], harmony search [48], and numerical iterative procedure [12] to deal with the complex instances (obstacles environment) have some notable drawbacks: (1) they may fail to find the optimal path, (2) they have limited capabilities when handling cases where the limits of maximum acceleration and maximum deceleration along the solution curve are no longer met, and (3) singular points or critical points of robot configuration may exist. Evolutionary algorithms offer an interesting alternative to overcome the above mentioned drawbacks [3]. Among the advantages of evolutionary techniques are:

– Population based search, and therefore it is more likely to avoid local minima,
– No auxiliary information such as gradients or derivatives is required for the algorithm itself,
– Over iterations and random selection, complex and multimodal problems can converge for global optimality, and
– Suitable to any kind of problem due to their problem independent nature [42].

To take advantage of these benefits, Saravanan et al. proposed a direct method called non-dominated sorting genetic algorithm to solve the minimum time trajectory with payload constraints for industrial robots in the presence of obstacles. They used cubic splines for the interpolation [42,44]. Hang et al. proposed a genetic algorithm procedure to solve point-to-point minimum jerk problem with fixed execution time [18]. In [5,27], a genetic algorithm procedure combined with a deterministic one based on interval analysis is used for global minimum time optimization.

Due to the important demand in maximizing the productivity of robotized operations, planning a minimum time trajectory a priori is a must. For such goal, this paper proposes a new evolutionary approach to solve the trajectory planning problem for industrial robots. The main contributions of this paper are:

- An evolutionary approach to solve the trajectory planning problem. This approach uses multiple populations genetic algorithm to obtain minimum time trajectories clamped with cubic splines. The use of multiple populations has been proved to be advantageous in other application and in our paper we extend it to the field of indirect robot trajectory planning. Moreover, the algorithm is independent of the degree of freedom (Dof) of the robot and of the robot type, and can work with different types (and combinations) of constraints including obstacle avoidance, kinematics, and dynamics constraints.
- An extensive statistical evaluation to show the effectiveness of the proposed approach as well as to show the improvement achieved by the proposed algorithm over the existing ones is performed. In addition, this evaluation validates the efficiency of GA algorithms in solving such problems.

This evolutionary approach is composed of two parallel populations genetic algorithms (PPGA). These algorithms are distinguished by means of boundary conditions applied to the system as shown in Section 2. The first algorithm, PPGA1, resolves the cubic polynomial joint trajectory formulation introduced by Lin et al. [25], where only kinematic constraints are considered. To illustrate the advantages of the PPGA1, a comparison is made with the results obtained from the application of an earlier version of a genetic algorithm procedure reported in [50]. Moreover, our first proposed procedure is extended to cater for dynamic constraints, such as, torques, power, and energy consumptions. In addition, a complete statistical evaluation have been established to evaluate the effectiveness of the proposed evolutionary algorithm. The second improved algorithm PPGA2 resolves the clamped cubic spline algorithm, where only velocities are nil at the initial and final configuration of the robot manipulator. The algorithm takes in consideration both kinematic and dynamic constraints.

2 Formulation of cubic polynomial joint trajectory

The philosophy of splining is to use low order polynomials to interpolate from grid point to grid point. This is ideally suited when one has control of the grid locations and the values of data being interpolated. As this control is dominated, the relative accuracy can be controlled by changing the overall space between the grid points.

Cubic splines are the lowest order polynomial endowed with inflection points. If one would think about interpolating a set of data points using parabolic functions without inflection points, the interpolation would be meaningless.

The path is given as a sequence of via-points in the operative space representing the position and the orientation of the robot end effector, and it is then transformed into a sequence of via-points in the joint space, by means of a kinematic inversion. The considered algorithm find an optimal trade-off execution time and, unlike most minimum time trajectory planning techniques found in the literature, it does not require to impose the execution time a priori. Constraints on the robot joints, such as upper bounds on velocity, acceleration, and jerk, are taken into account while executing the algorithms.

The formulation of the cubic spline is based on the n joint vectors (n configurations) that construct the joint trajectory. Joint vectors are denoted as q_i^j which represents the position of the joint i with respect to configuration j. The cubic polynomial trajectory is then constructed for each joint to fit the joint sequence q_i^0, $q_i^1, \cdots, q_i^n$. Let $t_0 < t_1 < \cdots < t_{n-2} < t_{n-1} < t_n$ be an ordered time sequence, at time $t = t_j$ the joint position will be q_i^j. Let $Q_i^j(t)$ be a cubic polynomial function defined on the time interval $[t_j, t_{(j+1)}]$; $0 \le j \le n-1$. The problem of trajectory interpolation is to spline $Q_i(t)$, for $i = 1, 2, \cdots, n-1$, together such that the required displacement, velocity, and acceleration are satisfied; and the displacement, velocity, and acceleration are continuous on the entire time interval $[t_1, t_n]$. Given that $Q_i(t)$ is cubic and represents the joint position, let $Q_i'(t)$ and $Q_i''(t)$ be the joint velocity and acceleration between q_i and q_{i+1}.

2.1 PPGA1 Formulation

In this formulation, boundary conditions are defined for both the velocity and acceleration at the initial and final points. The boundary conditions in this case are $\dot{q}_1 = \dot{q}_n = \ddot{q}_1 = \ddot{q}_n = 0$. Unlike common boundary conditions for cubic splines where either velocity (clamped boundary conditions) or acceleration (natural boundary conditions) are defined at end points, with constraints on both, constructing a spline requires extra Dof. To achieve this, two of the knots are not fixed. Thus, in this formulation, the n knots correspond to (n - 2) fixed input knots and 2 extra knots added to solve the spline system and their positions are solved as part of the linear system [25].

$$\begin{aligned} Q_i(t) = &\tfrac{Q_i''(t_i)}{6h_i}(t_{i+1}-t)^3 + \tfrac{Q_i''(t_{i+1})}{6h_i}(t-t_i)^3 \\ &+ \left[\tfrac{q_{i+1}}{h_i} - \tfrac{h_i Q_i''(t_{i+1})}{6}\right](t-t_i) \\ &+ \left[\tfrac{q_i}{h_i} - \tfrac{h_i Q_i''(t_i)}{6}\right](t_{i+1}-t) \end{aligned} \quad (1)$$

where $i = 1, 2, \cdots, n-1$, $h_i = t_{i+1} - t_i$

$$\begin{aligned} Q_i'(t) = &\tfrac{-Q_i''(t_i)}{2h_i}(t_{i+1}-t)^2 + \tfrac{Q_i''(t_{i+1})}{2h_i}(tt_i)^2 \\ &+ \left(\tfrac{q_{i+1}}{h_i} - \tfrac{h_i Q_i''(t_{i+1})}{6}\right) - \left(\tfrac{q_i}{h_i} - \tfrac{h_i Q_i''(t_i)}{6}\right) \end{aligned} \quad (2)$$

$$Q_i''(t) = \frac{t_{i+1}-t}{h_i} Q_i''(t_i) + \frac{(tt_i)}{h_i} Q_i''(t_{i-1}) \quad (3)$$

The two extra knots q_2 and q_{i-1} are not fixed and are used to add two new equations to the system in such a way that it can be solved. The joint positions of these two knots are

$$q_2 = q_1 + h_1\dot{q}_1 + \frac{h_1^2}{3}\ddot{q}_1 + \frac{h_1^2}{6}Q_1''(t_2) \quad (4)$$

$$q_{n-1} = q_n - h_{n-1}\dot{q}_n + \frac{h_{n-1}^2}{3}\ddot{q}_n + \frac{h_{n-1}^2}{6}Q_{n-2}''(t_{n-1}) \quad (5)$$

Using a continuity conditions on velocities and accelerations, a system of $n-2$ linear equations solving for unknowns $Q_i''(t_i)$'s is derived as [25]:

$$\mathbf{A}\begin{bmatrix} Q_2''(t_2) \\ Q_3''(t_3) \\ \vdots \\ Q_{n-1}''(t_{n-1}) \end{bmatrix} = \mathbf{Y} \quad (6)$$

where matrices $\mathbf{A}$ and $\mathbf{Y}$ are given by:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & & & & & \\ a_{21} & a_{22} & a_{23} & & & & \\ & a_{32} & a_{33} & a_{34} & & & \\ & & & \vdots & & & \\ & & & & a_{n-3,n-4} & a_{n-3,n-3} & a_{n-3,n-2} \\ & & & & & a_{n-2,n-3} & a_{n-2,n-2} \end{bmatrix}, \quad \mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ y_i \\ \vdots \\ y_{n-3} \\ y_{n-2} \end{bmatrix} \quad (7)$$

The values of a_{ij} and y_i are given in the Appendix. A unique solution is guaranteed once matrix $\mathbf{A}$ is non-singular.

2.2 PPGA2 Formulation

In the clamped cubic spline, the border conditions are $\dot{q}_1 = \dot{q}_n = 0$.

$$Q_i(t) = a_i + b_i(t-t_i) + c_i(t-t_i)^2 + d_i(t-t_i)^3, \quad (8)$$

$$Q_i'(t) = b_i + 2c_i(t-t_i) + 3d_i(t-t_i)^2, \quad (9)$$

$$Q_i''(t) = 2c_i + 6d_i(t-t_i) \quad (10)$$

where $i = 1, 2, \cdots, n-1$, $h_i = t_{i+1} - t_i$. A system of $n-2$ linear equations will be solved as following:

$$\mathbf{A}\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{bmatrix} = \mathbf{Y} \quad (11)$$

where

$$\mathbf{A} = \begin{bmatrix} 3h_1 & h_1 & & & & & \\ h_1 & 2(h_1+h_2) & h_1 & & 0 & & \\ & h_2 & 2(h_2+h_3) & h_3 & & & \\ & & & \vdots & & & \\ & 0 & & & h_{n-2} & 2(h_{n-2}+hn-1) & h_{n-1} \\ & & & & & h_{n-1} & 2h_{n-1} \end{bmatrix}, \quad (12)$$

and, matrix $\mathbf{Y}$ is given by:

$$\mathbf{Y} = \begin{bmatrix} \frac{3}{h_1}(a_2 - a_1) - 3\dot{q}_1 \\ \frac{3}{h_2}(a_3 - a_2) - \frac{3}{h_1}(a_2 - a_1) \\ \vdots \\ \frac{3}{h_{n-2}}(a_{n-1} - a_{n-2}) - \frac{3}{h_{n-3}}(a_{n-2} - a_{n-3}) \\ 3\dot{q}_{n-1} - \frac{3}{h_{n-1}}(a_n - a_{n-1}) \end{bmatrix} \tag{13}$$

3 Optimization technique using GA

The objective of this optimization procedure is to determine a set of optimum values of time intervals t_1, t_2, $\cdots$, t_{n-1}. A genetic algorithm procedure with parallel populations with migration technique has been implemented to optimize the time intervals needed to move the robot through a sequence of configurations [1]. This GA has multiple, independent populations. Each population evolves using steady-state genetic algorithm, but at each generation, some individuals migrate from one population to another. The migration algorithm is deterministic stepping-stone, where each population migrates a fixed number of its best individuals to its neighbor. The master population is updated each generation with the best individual from each population. The steady state genetic algorithm (SSGA) uses overlapping populations which gives the ability to specify how much of the population should be replaced in each generation. Newly generated offspring are added to the population, and then the worst individuals are destroyed.

Objective function

$$\text{Minimize} \sum_{i=1}^{n-1} h_i \tag{14}$$

subject to

1 Kinematic constraints

$$\begin{cases} \text{Joint positions:} & \left|q_i^j(t)\right| \leq q_i^{\max} \\ \text{Joint velocities:} & \left|\dot{q}_i^j(t)\right| \leq \dot{q}_i^{\max} \\ \text{Joint accelerations:} & \left|\ddot{q}_i^j(t)\right| \leq \ddot{q}_i^{\max} \\ \text{Joint jerks:} & \left|\dddot{q}_i^j(t)\right| \leq \dddot{q}_i^{\max} \end{cases} \tag{15}$$

2 Dynamic constraints

$$\begin{cases} \text{Joint torques: } |\tau_i(t)| \leq \tau_i^{\max} \\ \text{Joint power: } \ |P_i(t)| \leq P_i^{\max} \\ \text{Joint energy: } \ |E_i(t)| \leq E_i^{\max} \end{cases} \tag{16}$$

3 Payload constraints

$$Fg_{\min} \leq F_k \leq Fg_{\max} \ k = 1, 2 \tag{17}$$

where: $i = 1, \cdots, \text{DoF}$ of the robot, j takes the values from 1 to the number of nodes in the trajectory, F_k is the grasping force needed for a suitable static equilibrium during a grasping with two fingers gripper, $Fg_{\min} = 0$ and $Fg_{\max} = 60$ N are the minimum and maximum grasping forces used in the industrial application example, as explained in Section 4.3. This payload constraints was adopted earlier in [39,43]. The grasped object mass (payload) used in the illustration is 1 kg.

For industrial applications, the speed of operation affects the productivity. In order to maximize the speed of operation, the traveling time for the robot should be minimized. Thus, the optimization problem is to adjust the time intervals between each pair of adjacent configurations such that the total traveling time is minimal.

Chromosome consists of set of genes. Each gene contains a real number that represents the time interval. The number of genes depends on the fed path.

The value of each gene is selected randomly from $\left[t_j^{\min}, t_j^{\max}\right]$. The value of $t_j^{\max}$ will change in each generation depending on the new generated offsprings.

Selection A roulette-wheel selection method is applied.

Crossover The crossover operator defines the procedure for generating a child from two selected parents. The new child will be calculated as follows:

$$a_i = (mom_i + dad_i)/2, \tag{18}$$

$$bro_i = a_i \cdot dad_i + (1 - a_i) \cdot mom_i, \tag{19}$$

$$sis_i = a_i \cdot mom_i + (1 - a_i) \cdot dad_i \tag{20}$$

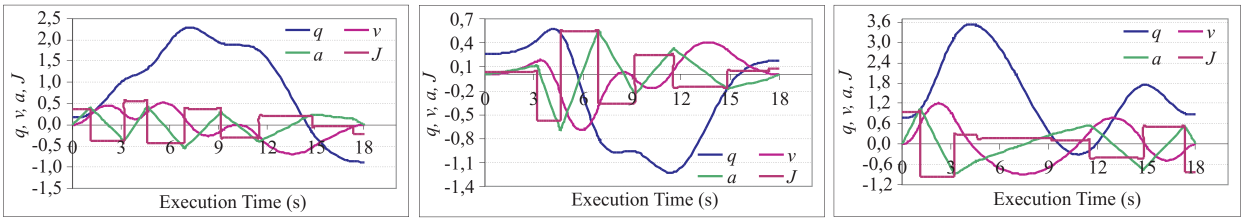
Mutation In this procedure, an offspring will be selected randomly then the algorithm will select a random set of genes in each chromosome for the mutation.

$$\begin{aligned} gene_j = gene_j + RV\left(t_j^{\min}, t_j^{\max}\right)[RV(t_j^{\min}, t_j^{\max}) \\ -RV(t_j^{\min}, t_j^{\max})] \end{aligned} \tag{21}$$

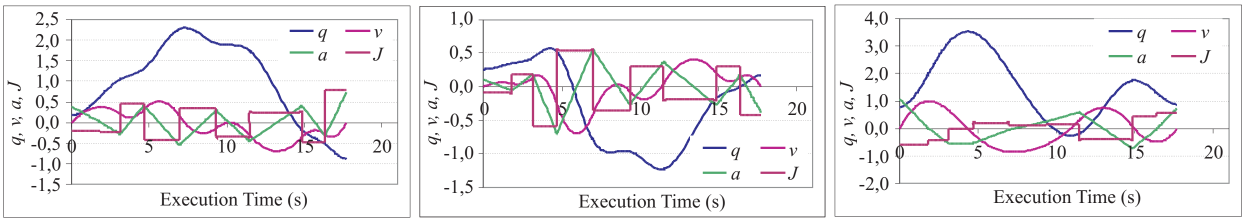
where $RV\left(t_j^{\min}, t_j^{\max}\right)$ means Random Value between $t_j^{\min}$ and $t_j^{\max}$.

4 Application Examples

The PPGA1 and PPGA2 described in this paper have been implemented using an Object Oriented C++ and tested in simulation for 6 DoF robot. The algorithms have been executed in a computer with Intel Core i5 – 2400 CPU 3.10 GHz, 4 GB of RAM. For the GA, the MIT GA Library [52] is used and adapted to the problem. In the following subsection, four different experiments are described to evaluate the efficiency of the proposed algorithms.



(a) PPGA1



(b) PPGA2

Fig. 1: 1st, 2nd, and 3rd joints: q = Positions [rad], v = Velocities [rad/s], a = Accelerations [rad/s^2], and J = Jerks [rad/s^3].

Table 1: Minimum time when $pc = 0.95$ with different values of pm

Mutation pm	Execution time (s)				Avg. Comp. time (s)
	Results in [50]	Min in PPGA1	Avg. in PPGA1	PPGA2	
0.001	20.156	17.657	18.716	18.091	0.355
0.01	19.880	17.223	17.824	17.726	0.402
0.05	18.211	17.145	17.617	17.706	0.836
0.1	18.226	17.138	17.522	17.971	1.401
0.2	18.929	17.221	17.622	17.896	1.732
0.3	18.957	17.339	17.689	17.897	2.062
0.4	19.062	17.362	17.846	17.931	2.562

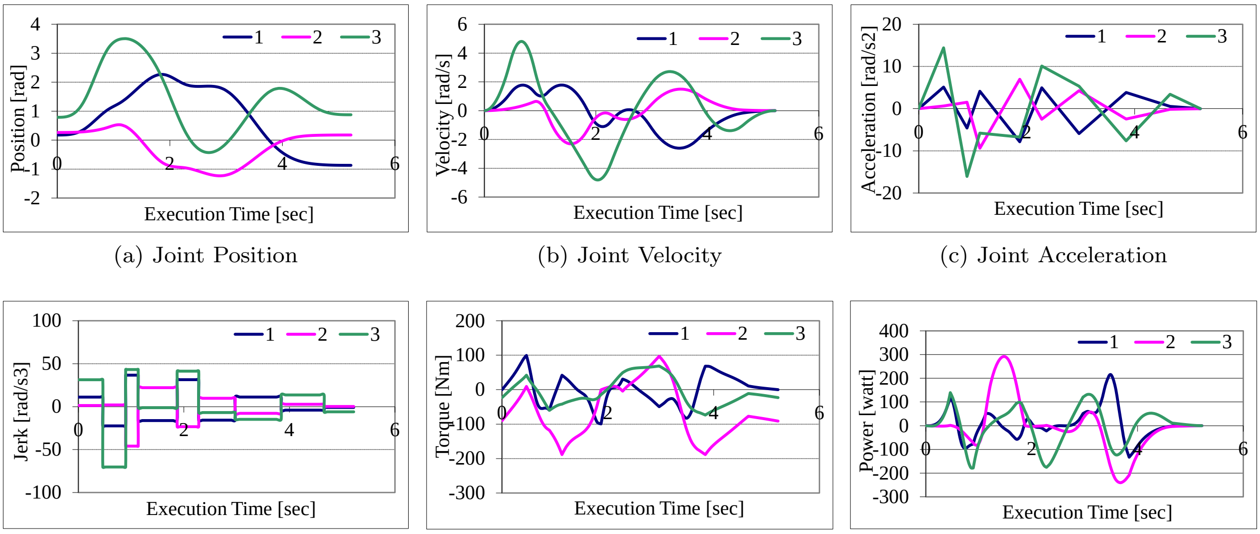
4.1 Experimenting with kinematic constraints

In this experiment, in order to compare the proposed procedure with previously reported results in literature, a PUMA560 robot is used. For more details about the example characteristics and data set, please refer to [25] and [50]. In this first experiment, only kinematic constraints are considered and resolved using PPGA1 & PPGA2. Many combinations of crossover and mutation probabilities are experimented to find the best combination that gives the best combination of minimum execution time and better computational time.

In Fig. 1, the smoothness of the joint position and velocity curves can be noticed. Moreover, the motion does not violate the kinematic limits. In this scope, multi runs of the algorithm are tested by changing the mutation probability and fixing the crossover probability to 0.95, see Table 1. The combination of $pc = 0.95$ and $pm = 0.05$ is chosen to do the rest of experiments as it produced the best combination of execution time and computational time. However, the combination $pc = 0.95$ and $pm = 0.1$ gives better execution time, but worse computation time. Observing the results in Table 1, the minimum time found using PPGA1 is 17.145 s, and for PPGA2 is 17.706 s at $pc = 0.95$ and $pm = 0.05$, while the minimum time obtained by [50] is 18.211 s, and by [25] is 18.451 s. Besides, the rest of results in Table 1 are better than the results reported in [50]. In addition, the results obtained when $pc = 0.35$, 0.65 and $pm = 0.01$ and 0.05 are 18.112 and 18.191 s for PPGA1, 18.087 and 18.009 s for PPGA2 respectively, which are better than Tse and Wang's [50] results which are 18.356 and 18.258 s.

4.2 Experimenting with kinematic and dynamic constraints

In this example, kinematics and dynamic constraints are considered and solved using PPGA2 with crossover probability $pc = 0.95$ and mutation probability $pm =$ 0.05. The sequence of configurations used is the same as



(a) Joint Position (b) Joint Velocity (c) Joint Acceleration

(d) Joint Jerk (e) Joint Torque (f) Joint Power

Fig. 2: Trajectory evolution for the first three joint solved by PPGA2 with kinematics and dynamics constraints.

the one used in the previous example (Section 4.1). The recursive Newton-Euler formulations are used to solve the inverse dynamic problem [9]. The robot parameters and dynamic limits are extracted from [7]. The minimum time is found to be 5.2187 s.

Observing Fig. 2, the motion does not violate the constraints. Moreover, the generated trajectories are smooth and the required torques for the motion approximate the set limits without violating them.

4.3 Industrial Application with payload

This experiment represents a simulation of a general configuration of a pick and place operation in complex industrial environment. The initial and final configurations (C^i and C^f) and obstacles dimensions and their positions are tabulated in Tables 2 and 3, respectively. In this example, the path-planning problem has been solved first by Abu-Dakka et al [2], and then the trajectory has been adjusted using PPGA2. In this case, the optimization problem is to move the robot with its payload from an initial configuration to a final one while optimizing the time considering the robot physical constraints, dynamic constraints, and the payload itself. The payload influence on the dynamic model of the manipulator is considered by adding the grasped object mass and the effect of its inertial moment to the last link, in addition to the fact that the payload grasping force is constrained according to Eq. 17. This constraint is responsible for maintaining the object firmly grasped. This indirect method of obtaining the trajectory has been compared with the direct method developed in [36] where three different algorithms were used (A*, UC = Uniform Cost, G = Gready).

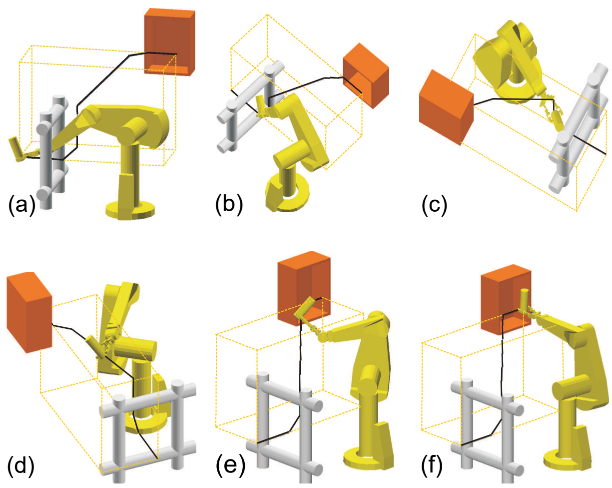


Fig. 3: The path and trajectory evolution for the industrial application example [2]. (a) and (f) represent the robot configurations at the pick and place positions of the object respectively. (b) to (e) represent intermediate configurations of the pick and place operation.

The illustrative manipulator task consists in transporting a payload from an initial to a final configuration. Four control parameters are used for comparison: d_s which is the distance between significant points, d_e is the distance between the initial and final positions of the end-effector, t_c is the computational time, and

Table 2: Initial C^i and Final C^f configurations for the industrial application example

	1	2	3	4	5	6
C^i	−7.50	−174.80	46.40	4.30	16.50	−6.50
C^f	−95.10	−101.20	15.59	0.00	0.00	0.00

t_e is the time needed to execute the trajectory. The evolution of the trajectory is illustrated in Fig. 3. Table 4 illustrates clearly that the presented algorithm; with even more constraints, provides an improvement over the results of the algorithms described in [36].

4.4 Statistical evaluation of the PPGA1

In this section, two sets of experiments have been implemented in order to statistically evaluate the proposed algorithms. In the first set, the problem data is extracted from [25] and [50], and already mentioned in Section 4.1. This statistical evaluation is based on 50,000 runs of the algorithm. The setup of this experiment consists of 500 replicates for each different combination between pc and pm. Figures 4, 5, 6, 7, 8, 9, 10, 11, 12, and 13 show the graphical representation of the variance test for these runs. For each pc, 5000 runs are executed divided in groups of 500 runs which correspond to different pm. From tables in Figs 4, 5, 6, 7, 8, 9, 10, 11, 12, and 13, it can be observed that the execution time is normally distributed (p-value < 0.05). The best value of the execution time obtained is 17.1316 s with computational cost 1.585 s, when $pc = 0.75$ and $pm = 0.09$.

In order to compare the performance of pc and pm, the statistical of test for the equality of means (analysis of variance ANOVA) is used. The null-hypothesis in ANOVA test states that the means of different levels of a parameter are equal, while they are not equal in the alternative hypothesis. Thus, it can be concluded that the parameter has an effect upon the response variable. In this experiment, the main effect involves the independent variables (pc and pm) one at a time. The effect of one factor on the other is called the interaction effect. For each hypothesis, there is a F value which is the mean square for each main effect and the interaction one divided by the within variance. The p-value is the probability to observe F value as large as we did under the null-hypothesis. These p-values in ANOVA test are only valid if the responses (execution time or computational time: the case of this paper) are normally distributed, which are already improved in Figs. 4, 5, 6, 7, 8, 9, 10, 11, 12, and 13 and 14, 15, 16, 17, 18, 19, 20, 21, 22, and 23. The decision that the parameter has an effect upon the response variable depends on the p-value if it is equal or less than a chosen level of significance. This level of significance in this paper is 0.05. The results of the ANOVA test of 500 replicates for 50,000 runs are tabulated in Tables 5 and 6 for both execution and computational time. In Table 5, crossover and mutation are both highly statistically significant on the execution time as the interaction between them with p-value $= 0.0004$. Table 6 also shows that crossover and mutation are both highly statistically significant on computational time as the interaction between them with p-value $= 0.0113$.

Figure 24 represents the average execution and computational time for each combination of pc and pm. The computational time values have been shifted in y-axis by 17 s to overlap with the execution time values, so it will be easy to study the results. Observing the figure leads us to use $pc = 0.95$ and $pm = 0.05$ to obtain the best combination of execution and computational time.

On the other hand, a set of data introduced by Gasparetto and Zanotto [17], is used for a comparative analysis between the proposed PPGA1 and five different techniques. These techniques are:

i the Sequential Quadratic Program (SQP) using cubic spline [17],
ii the Harmony Search algorithm (HS) [48],
iii interval analysis using cubic spline [33],
iv trigonometric spline [47], and
v synchronized trigonometric S-curve [30].

The data used for this evaluation are detailed in Table 7, and extracted from [17].

Gasparetto and Zanotto's algorithm [17] determined the initial interval time value and used SQP technique (the MATLAB function *fmincon*) for minimum time trajectory ($k_T = 1$, $k_J = 0$). However, they adjusted k_T and k_J so that the execution time is 9.1 s and the maximum jerk is $46.85°/s^3$ which is better than the jerk value in [33]. The HS simulation [48] gave a minimum time trajectory of 8.5718 s for 10,000 iterations and 8.5577 s for 200,000 iterations with maximum jerk close to $60°/s^3$ (plot information). Simon and Isik [47] achieved 9.1 s with maximum jerk $80.84°/s^3$ for the same set of data and using trigonometric spline, while 7.5398 s is obtained using S-curve [30] with maximum jerk $16.4178°/s^3$. However, no via points are used in [30], the planning was directly between the initial and final configurations. The PPGA1, in this paper, is achieving an average of 7.19 s execution time and maximum jerk $80.2°/s^3$. The PPGA1 is using the following GA parameters in Table 8:

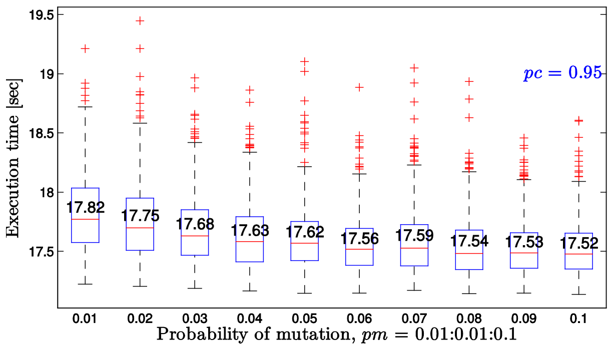
A detailed analysis of results for this comparison is tabulated in Table 9:

Table 3: Obstacles locations and characteristics where $C_i^{Cyl,1}$ and $C_i^{Cyl,2}$ are the centers of cylinder i with radius r_i^{Cyl}

1st Cylindrical obstacle	2nd Cylindrical obstacle	3rd Cylindrical obstacle	4th Cylindrical obstacle
$C_1^{Cyl,1} = (-0.7, 0.5, 0.0)$ $C_1^{Cyl,2} = (-0.7, 0.5, 0.8)$ $r_1^{Cyl} = 0.15$	$C_2^{Cyl,1} = (-0.7, 0.0, 0.0)$ $C_2^{Cyl,2} = (-0.7, 0.0, 0.8)$ $r_2^{Cyl} = 0.15$	$C_3^{Cyl,1} = (-0.7, -0.15, 0.7)$ $C_3^{Cyl,2} = (-0.7, 0.65, 0.7)$ $r_3^{Cyl} = 0.15$	$C_4^{Cyl,1} = (-0.7, -0.15, 0.15)$ $C_4^{Cyl,2} = (-0.7, 0.65, 0.15)$ $r_4^{Cyl} = 0.15$
1st Prismatic obstacle	**2nd Prismatic obstacle**	**3rd Prismatic obstacle**	**4th Prismatic obstacle**
$P_{11} = (0.31, 0.79, 1.42)$ $P_{12} = (0.31, 0.99, 1.42)$ $P_{13} = (0.31, 0.79, 0.97)$ $P_{14} = (0.31, 0.99, 0.97)$	$P_{21} = (0.31, 0.79, 1.42)$ $P_{22} = (0.31, 0.99, 1.42)$ $P_{23} = (-0.03, 0.99, 1.42)$ $P_{24} = (-0.03, 0.79, 1.42)$	$P_{31} = (-0.03, 0.79, 1.42)$ $P_{32} = (-0.03, 0.99, 1.42)$ $P_{33} = (-0.03, 0.99, 0.97)$ $P_{34} = (-0.03, 0.79, 0.97)$	$P_{41} = (-0.03, 0.79, 0.97)$ $P_{42} = (-0.03, 0.99, 0.97)$ $P_{43} = (0.31, 0.99, 0.97)$ $P_{44} = (0.31, 0.79, 0.97)$

Table 4: Performance comparison of the PPGA2

	Rubio's results [36]			Paper's
	A*	UC	G	results
$\boldsymbol{d_s}$ (m)	5.82	5.41	5.43	4.32
$\boldsymbol{d_e}$ (m)				1.79
$\boldsymbol{t_c}$ (s)	17,049	16,233	2674	17.78
$\boldsymbol{t_e}$ (s)	35.61	29.23	45.70	1.63



pm	runs	Mean[sec]	Std. Dev.	min [sec]	max [sec]
0.01	500	17.8236	0.32557	17.2230	19.2111
0.02	500	17.7547	0.33487	17.2043	19.4473
0.03	500	17.6834	0.30199	17.1872	18.9649
0.04	500	17.6301	0.28891	17.1665	18.8618
0.05	500	17.6165	0.27730	17.1450	19.1023
0.06	500	17.5614	0.24647	17.1474	18.8853
0.07	500	17.5875	0.29012	17.1709	19.0486
0.08	500	17.5401	0.26350	17.1422	18.9334
0.09	500	17.5325	0.24452	17.1470	18.4570
0.1	500	17.5217	0.23793	17.1376	18.6074
Pooled	5000	17.6251	0.28292		
Bartlett's statistic		127.561			
Degrees of freedom		9			
p-value		0			

Fig. 4: Execution time statistics when $pc = 0.95$ and pm varies from 0.01 to 0.1 with an interval of 0.01.

In general, each run of genetic algorithms produces a different solution because of the random nature of the calculation involved. In Section 4.1, it can be seen clearly that the results change by modifying the mutation probability, see Table 1. In this section, the genetic algorithm (PPGA1) uses the fixed set of parameters listed in Table 8. However, to statistically evaluate the proposed algorithm, 100 runs have been performed with different seed values to enrich the randomness. The statistical summary of these runs shows that even for the worst value of a trajectory execution time of 7.4413 s, this is still less than the results reported in other works cited earlier in this section. The minimum value achieved is 6.9726 s, while the average is 7.198 s. Figure 25 illustrates the statistical evaluation, where the upper graph shows the trajectories execution time with its average, standard deviation, etc., and the lower one shows the procedure computational time for each run and its average.

5 Conclusion

This paper proposes an evolutionary approach to solve the trajectory planning problem. This approach uses multiple populations genetic algorithm with migration technique to obtain minimum time trajectories clamped with cubic splines. The use of multiple populations has been proved to be advantageous in other application and in our paper we extend it to the field of indirect robot trajectory planning. This approach has the ability to combine different types of constraints; obstacle avoidance, kinematics, and dynamics. Moreover, it can be adapted to different manipulators since the model is independent of the type and number of DoF of the robot. A considerably large number of experiments (more than 50,000) have been done based on benchmark paths extracted from the literature. The results of these experiments are used in an extensive statistical evaluation to show the effectiveness of the proposed

Table 5: ANOVA test of 500 replicates (Execution time).

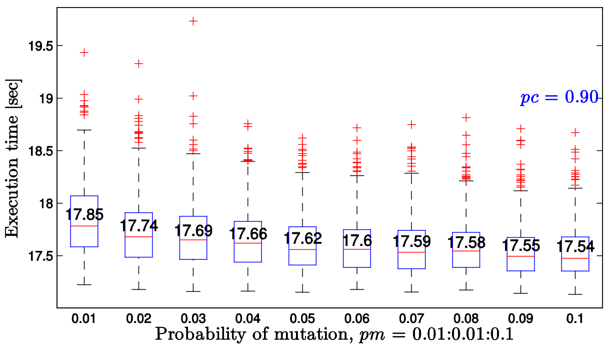
Source	Sum of Squares	df	Mean Squares	F-Value	p-value
Mutation probability	553.72	9	61.5245	592.31	0
Crossover probability	153.76	9	17.0842	164.47	0
Interaction	13.54	81	0.1672	1.61	0.0004
Error	5183.27	49900	0.1039		
Total	5904.29	49999			

Table 6: ANOVA test of 500 replicates (Computational time).

Source	Sum of Squares	df	Mean Squares	F-Value	p-value
Mutation probability	5387.7	9	598.633	33754.09	0
Crossover probability	0.38	9	0.042	2.37	0.0113
Interaction	3.99	81	0.049	2.78	0
Error	884.98	49900	0.018		
Total	6277.05	49999			

Table 7: Kinematic limits of the joints

Joint no.	1	2	3	4	5	6
Initial configuration (°)	-10	20	15	150	30	120
Virtual configuration	—	—	—	—	—	—
Via config. 1	60	50	100	100	110	60
Via config. 2	20	120	-10	40	90	100
Virtual configuration	—	—	—	—	—	—
Final configuration	55	35	30	10	70	25
Velocity $(^\circ/s)$	100	95	100	150	130	110
Acceleration $(^\circ/s^2)$	60	60	75	70	90	80
Jerk $(^\circ/s^3)$	60	66	85	70	75	70



pm	runs	Mean[sec]	Std. Dev.	min [sec]	max [sec]
0.01	500	17.8458	0.35512	17.2227	19.4337
0.02	500	17.7387	0.33933	17.1780	19.3286
0.03	500	17.6912	0.30735	17.1582	19.7326
0.04	500	17.6622	0.29470	17.1618	18.7567
0.05	500	17.6181	0.28230	17.1521	18.6213
0.06	500	17.5983	0.27079	17.1784	18.7170
0.07	500	17.5900	0.28219	17.1537	18.7479
0.08	500	17.5795	0.26075	17.1720	18.8147
0.09	500	17.5505	0.26261	17.1414	18.7081
0.1	500	17.5405	0.25593	17.1324	18.6719
Pooled	5000	17.6415	0.29286		
Bartlett's statistic	117.248				
Degrees of freedom	9				
p-value	0				

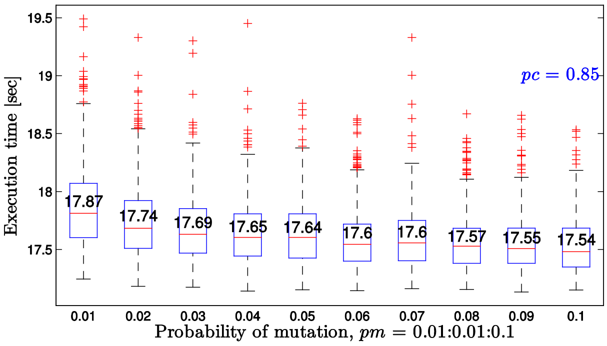
Fig. 5: Execution time statistics when $pc = 0.90$ and pm varies from 0.01 to 0.1 with an interval of 0.01.

Table 8: PPGA1 parameters

Parameter	Value
Population size	30
Number of populations	3
Number of migrations	15
Number of generations	80
Crossover probability	0.95
Mutation probability	0.05
% of solutions replaced by new generation	30 %

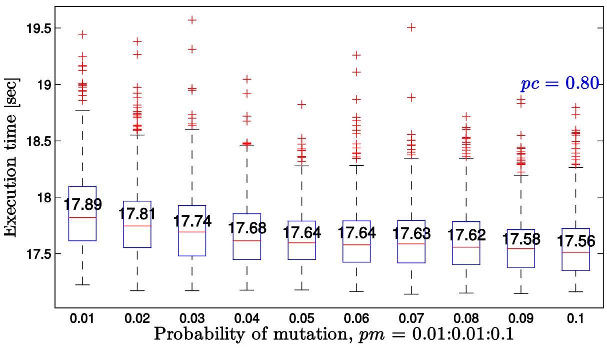
Table 9: Execution time comparison for the proposed PPGA1

	Trajectory generation scheme	Execution time (sec)
Gasparetto and Zanotto [17]	Cubic spline	8.5726
Tangpattanakul,and Artrit [48]	Cubic spline	8.5577
Simon and Isik [47]	Trigonometric spline	9.1
Piazzi, and Visioli [33]	Cubic spline	9.1
Perumaal, and Jawahar [30]	Synchronized trigonometric S-curve	7.5398
Proposed PPGA1	**Cubic spline**	**6.97259**



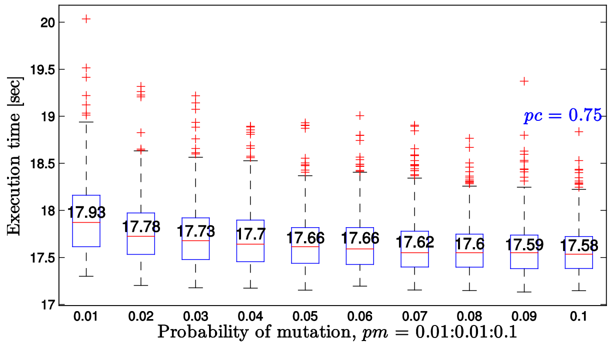
pm	runs	Mean[sec]	Std. Dev.	min [sec]	max [sec]
0.01	500	17.8673	0.36153	17.2447	19.4907
0.02	500	17.7426	0.33527	17.1831	19.3294
0.03	500	17.6854	0.30418	17.1737	19.3010
0.04	500	17.6492	0.28489	17.1407	19.4507
0.05	500	17.6426	0.28897	17.1518	18.7619
0.06	500	17.5958	0.27635	17.1440	18.6283
0.07	500	17.6032	0.28086	17.1619	19.3292
0.08	500	17.5656	0.25750	17.1547	18.6705
0.09	500	17.5535	0.25620	17.1330	18.6573
0.1	500	17.5360	0.24848	17.1515	18.5329
Pooled	5000	17.6441			
Bartlett's statistic		134.017			
Degrees of freedom		9			
p-value		0			

Fig. 6: Execution time statistics when $pc = 0.85$ and pm varies from 0.01 to 0.1 with an interval of 0.01.



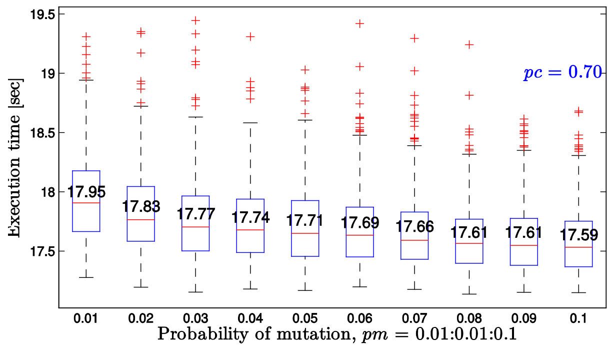
pm	runs	Mean[sec]	Std. Dev.	min [sec]	max [sec]
0.01	500	17.8900	0.37812	17.2238	19.4408
0.02	500	17.8063	0.35182	17.1729	19.3802
0.03	500	17.7411	0.33410	17.1716	19.5695
0.04	500	17.6804	0.30129	17.1762	19.0460
0.05	500	17.6406	0.26702	17.1793	18.8210
0.06	500	17.6381	0.30512	17.1653	19.2588
0.07	500	17.6319	0.29231	17.1415	19.5051
0.08	500	17.6194	0.29086	17.1512	18.7139
0.09	500	17.5824	0.27130	17.1486	18.8674
0.1	500	17.5646	0.27970	17.1618	18.7954
Pooled	5000	17.6795	0.30910		
Bartlett's statistic		122.836			
Degrees of freedom		9			
p-value		0			

Fig. 7: Execution time statistics when $pc = 0.80$ and pm varies from 0.01 to 0.1 with an interval of 0.01.



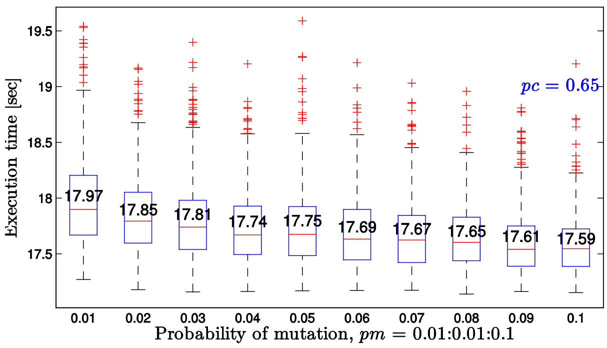
pm	runs	Mean[sec]	Std. Dev.	min [sec]	max [sec]
0.01	500	17.9330	0.39452	17.2995	20.0360
0.02	500	17.7805	0.33647	17.2015	19.3169
0.03	500	17.7325	0.33489	17.1765	19.2183
0.04	500	17.7020	0.33214	17.1742	18.8941
0.05	500	17.6603	0.30744	17.1506	18.9301
0.06	500	17.6593	0.31162	17.1949	19.0071
0.07	500	17.6171	0.30644	17.1534	18.9054
0.08	500	17.5978	0.26957	17.1467	18.7649
0.09	500	17.5882	0.27739	17.1316	19.3729
0.1	500	17.5800	0.26297	17.1452	18.8368
Pooled	5000	17.6851	0.31554		
Bartlett's statistic		137.408			
Degrees of freedom		9			
p-value		0			

Fig. 8: Execution time statistics when $pc = 0.75$ and pm varies from 0.01 to 0.1 with an interval of 0.01.



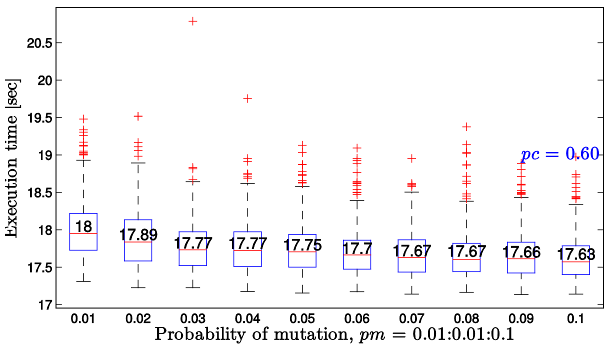
pm	runs	Mean[sec]	Std. Dev.	min [sec]	max [sec]
0.01	500	17.9518	0.38090	17.2766	19.3087
0.02	500	17.8293	0.35104	17.1949	19.3509
0.03	500	17.7656	0.35880	17.1536	19.4455
0.04	500	17.7357	0.32246	17.1793	19.3081
0.05	500	17.7114	0.34753	17.1676	19.0280
0.06	500	17.6918	0.31851	17.1977	19.4188
0.07	500	17.6602	0.31556	17.1767	19.2945
0.08	500	17.6097	0.28689	17.1368	19.2409
0.09	500	17.6095	0.29950	17.1519	18.6143
0.1	500	17.5887	0.28149	17.1489	18.6808
Pooled	5000	17.7154	0.32773		
Bartlett's statistic		89.362			
Degrees of freedom		9			
p-value		0			

Fig. 9: Execution time statistics when $pc = 0.70$ and pm varies from 0.01 to 0.1 with an interval of 0.01.



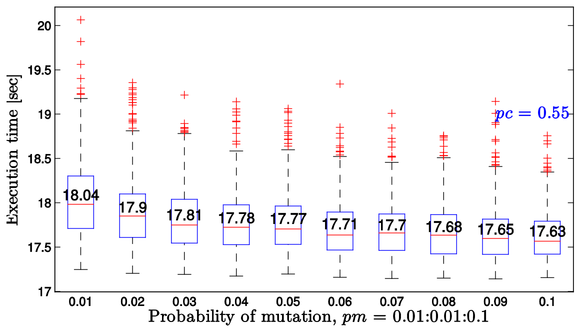
pm	runs	Mean[sec]	Std. Dev.	min [sec]	max [sec]
0.01	500	17.9708	0.40476	17.2690	19.5408
0.02	500	17.8468	0.35249	17.1772	19.1666
0.03	500	17.8054	0.37302	17.1594	19.3977
0.04	500	17.7407	0.34113	17.1608	19.2040
0.05	500	17.7473	0.36679	17.1686	19.5904
0.06	500	17.6922	0.33083	17.1707	19.2146
0.07	500	17.6745	0.32852	17.1740	19.0308
0.08	500	17.6537	0.29835	17.1392	18.9570
0.09	500	17.6071	0.30277	17.1610	18.8086
0.1	500	17.5863	0.27436	17.1510	19.2050
Pooled	5000	17.7325	0.33933		
Bartlett's statistic		120.547			
Degrees of freedom		9			
p-value		0			

Fig. 10: Execution time statistics when $pc = 0.65$ and pm varies from 0.01 to 0.1 with an interval of 0.01.



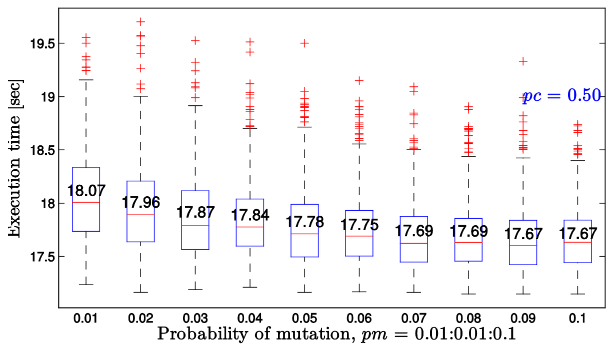
pm	runs	Mean[sec]	Std. Dev.	min [sec]	max [sec]
0.01	500	17.9969	0.39304	17.3106	19.4802
0.02	500	17.8873	0.39817	17.2242	19.5172
0.03	500	17.7748	0.34664	17.2238	20.7859
0.04	500	17.7711	0.34815	17.1778	19.7525
0.05	500	17.7500	0.33486	17.1554	19.1288
0.06	500	17.7014	0.30602	17.1702	19.0913
0.07	500	17.6720	0.30315	17.1409	18.9516
0.08	500	17.6657	0.32711	17.1641	19.3763
0.09	500	17.6557	0.30552	17.1353	18.8862
0.1	500	17.6266	0.29650	17.1405	18.9736
Pooled	5000	17.7502	0.33769		
Bartlett's statistic		103.233			
Degrees of freedom		9			
p-value		0			

Fig. 11: Execution time statistics when $pc = 0.60$ and pm varies from 0.01 to 0.1 with an interval of 0.01.



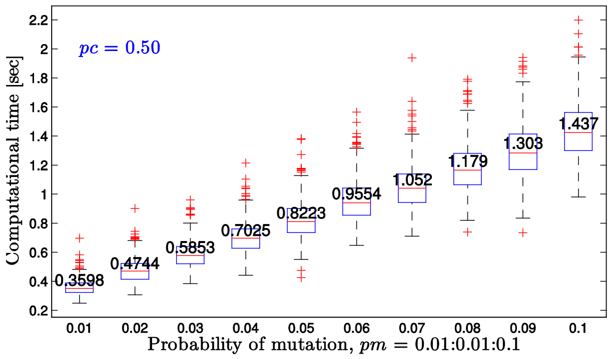
pm	runs	Mean[sec]	Std. Dev.	min [sec]	max [sec]
0.01	500	18.0386	0.45081	17.2460	20.0629
0.02	500	17.8991	0.40378	17.2034	19.3554
0.03	500	17.8140	0.36323	17.1911	19.2162
0.04	500	17.7798	0.35070	17.1728	19.1399
0.05	500	17.7714	0.33565	17.1960	19.0603
0.06	500	17.7112	0.33089	17.1588	19.3400
0.07	500	17.6994	0.31249	17.1449	19.0088
0.08	500	17.6801	0.32170	17.1481	18.7556
0.09	500	17.6509	0.31512	17.1419	19.1426
0.1	500	17.6293	0.27998	17.1542	18.7563
Pooled	5000	17.7674	0.34960		
Bartlett's statistic		176.691			
Degrees of freedom		9			
p-value		0			

Fig. 12: Execution time statistics when $pc = 0.55$ and pm varies from 0.01 to 0.1 with an interval of 0.01.



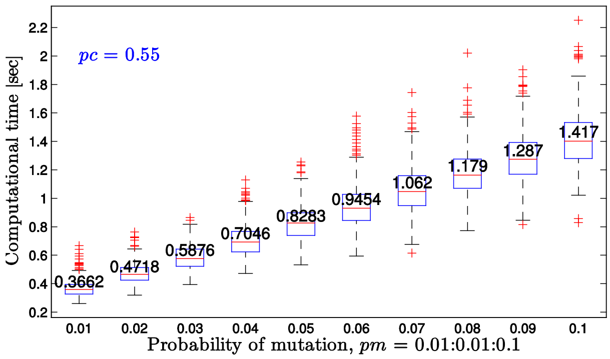
pm	runs	Mean[sec]	Std. Dev.	min [sec]	max [sec]
0.01	500	18.0706	0.43716	17.2359	19.5537
0.02	500	17.9613	0.43266	17.1639	19.7019
0.03	500	17.8686	0.40120	17.1895	19.5250
0.04	500	17.8422	0.37250	17.2112	19.5115
0.05	500	17.7780	0.37575	17.1625	19.5001
0.06	500	17.7506	0.34449	17.1687	19.1491
0.07	500	17.6943	0.33533	17.1625	19.0914
0.08	500	17.6902	0.32123	17.1472	18.9049
0.09	500	17.6652	0.31825	17.1485	19.3304
0.1	500	17.6672	0.29336	17.1479	18.7389
Pooled	5000	17.7988	0.36618		
Bartlett's statistic		163.29			
Degrees of freedom		9			
p-value		0			

Fig. 13: Execution time statistics when $pc = 0.50$ and pm varies from 0.01 to 0.1 with an interval of 0.01.



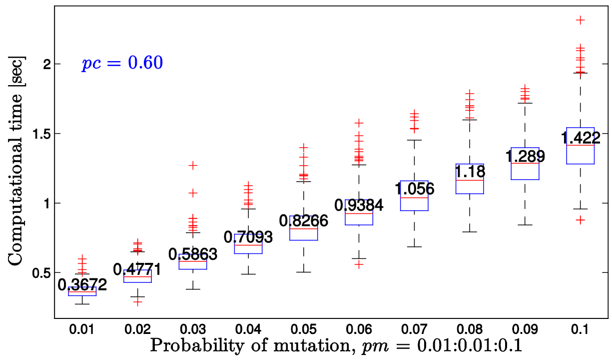
pm	runs	Mean[sec]	Std. Dev.	min [sec]	max [sec]
0.01	500	0.35981	0.05197	0.250	0.697
0.02	500	0.47444	0.08007	0.307	0.901
0.03	500	0.58534	0.09026	0.384	0.961
0.04	500	0.70247	0.10833	0.442	1.214
0.05	500	0.82230	0.12995	0.426	1.382
0.06	500	0.95537	0.14430	0.648	1.565
0.07	500	1.05173	0.15916	0.710	1.939
0.08	500	1.17901	0.16573	0.739	1.791
0.09	500	1.30280	0.19302	0.734	1.940
0.1	500	1.43729	0.19310	0.980	2.198
Pooled	5000	0.88705	0.13930		
Bartlett's statistic		1296.01			
Degrees of freedom		9			
p-value		0			

Fig. 14: Computational time statistics when $pc = 0.50$ and pm varies from 0.01 to 0.1 with an interval of 0.01.



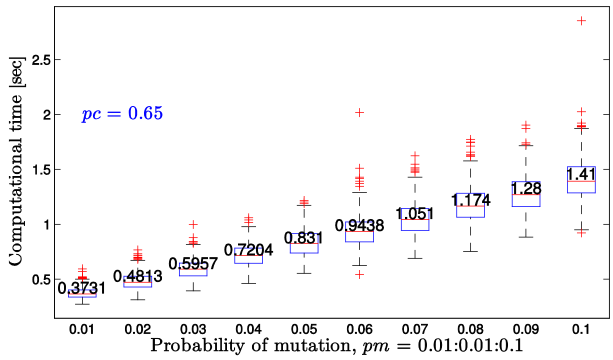
pm	runs	Mean[sec]	Std. Dev.	min [sec]	max [sec]
0.01	500	0.36619	0.05669	0.261	0.667
0.02	500	0.47183	0.06805	0.320	0.764
0.03	500	0.58763	0.08769	0.393	0.865
0.04	500	0.70458	0.11074	0.472	1.129
0.05	500	0.82827	0.12133	0.533	1.255
0.06	500	0.94539	0.15055	0.594	1.578
0.07	500	1.06193	0.16519	0.615	1.743
0.08	500	1.17887	0.16512	0.773	2.021
0.09	500	1.28713	0.17183	0.815	1.903
0.1	500	1.41695	0.19825	0.828	2.251
Pooled	5000	0.88488	0.13732		
Bartlett's statistic		1317.97			
Degrees of freedom		9			
p-value		0			

Fig. 15: Computational time statistics when $pc = 0.55$ and pm varies from 0.01 to 0.1 with an interval of 0.01.



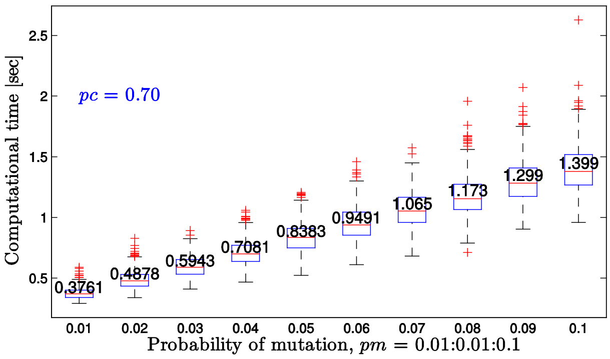
pm	runs	Mean[sec]	Std. Dev.	min [sec]	max [sec]
0.01	500	0.36720	0.04713	0.274	0.599
0.02	500	0.47714	0.06590	0.290	0.715
0.03	500	0.58625	0.09161	0.381	1.270
0.04	500	0.70926	0.10604	0.489	1.125
0.05	500	0.82658	0.12884	0.504	1.398
0.06	500	0.93836	0.14268	0.559	1.575
0.07	500	1.05634	0.15462	0.686	1.642
0.08	500	1.18046	0.16647	0.792	1.786
0.09	500	1.28871	0.17126	0.843	1.822
0.1	500	1.42174	0.20494	0.878	2.315
Pooled	5000	0.88520	0.13642		
Bartlett's statistic		1490.28			
Degrees of freedom		9			
p-value		0			

Fig. 16: Computational time statistics when $pc = 0.60$ and pm varies from 0.01 to 0.1 with an interval of 0.01.



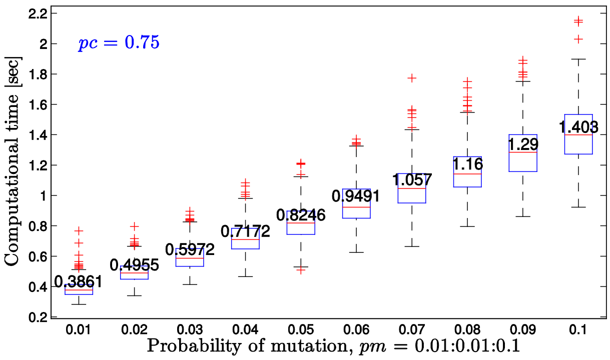
pm	runs	Mean[sec]	Std. Dev.	min [sec]	max [sec]
0.01	500	0.37312	0.04940	0.273	0.594
0.02	500	0.48128	0.07346	0.311	0.767
0.03	500	0.59596	0.08809	0.394	0.998
0.04	500	0.72044	0.10656	0.462	1.059
0.05	500	0.83098	0.12848	0.553	1.217
0.06	500	0.94384	0.14904	0.543	2.018
0.07	500	1.05056	0.15348	0.690	1.624
0.08	500	1.17425	0.16543	0.754	1.774
0.09	500	1.28041	0.16850	0.883	1.904
0.1	500	1.41005	0.19952	0.922	2.854
Pooled	5000	0.88606	0.13598		
Bartlett's statistic		1350.47			
Degrees of freedom		9			
p-value		0			

Fig. 17: Computational time statistics when $pc = 0.65$ and pm varies from 0.01 to 0.1 with an interval of 0.01.



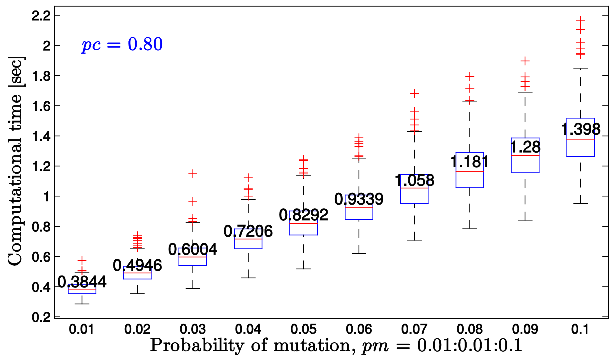
pm	runs	Mean[sec]	Std. Dev.	min [sec]	max [sec]
0.01	500	0.37609	0.04893	0.292	0.589
0.02	500	0.48777	0.07556	0.339	0.828
0.03	500	0.59432	0.08402	0.410	0.892
0.04	500	0.70807	0.10260	0.468	1.059
0.05	500	0.83834	0.12319	0.523	1.206
0.06	500	0.94907	0.13736	0.610	1.459
0.07	500	1.06473	0.15044	0.682	1.573
0.08	500	1.17348	0.16805	0.712	1.957
0.09	500	1.29911	0.18001	0.903	2.070
0.1	500	1.39899	0.19592	0.959	2.627
Pooled	5000	0.88900	0.13468		
Bartlett's statistic		1403.15			
Degrees of freedom		9			
p-value		0			

Fig. 18: Computational time statistics when $pc = 0.70$ and pm varies from 0.01 to 0.1 with an interval of 0.01.



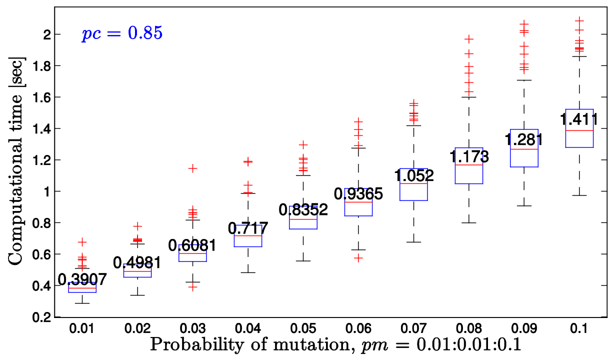
pm	runs	Mean[sec]	Std. Dev.	min [sec]	max [sec]
0.01	500	0.38614	0.05239	0.283	0.767
0.02	500	0.49551	0.06797	0.340	0.796
0.03	500	0.59715	0.08497	0.413	0.896
0.04	500	0.71718	0.10222	0.465	1.084
0.05	500	0.82457	0.11717	0.509	1.213
0.06	500	0.94905	0.13701	0.626	1.371
0.07	500	1.05663	0.14947	0.664	1.773
0.08	500	1.16016	0.15562	0.796	1.750
0.09	500	1.29016	0.1802	0.862	1.891
0.1	500	1.40349	0.19361	0.923	2.154
Pooled	5000	0.88800	0.13193		
Bartlett's statistic		1373.4			
Degrees of freedom		9			
p-value		0			

Fig. 19: Computational time statistics when $pc = 0.75$ and pm varies from 0.01 to 0.1 with an interval of 0.01.



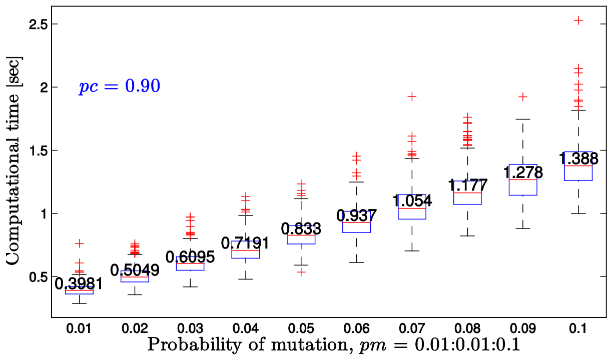
pm	runs	Mean[sec]	Std. Dev.	min [sec]	max [sec]
0.01	500	0.38444	0.04444	0.285	0.573
0.02	500	0.49464	0.06620	0.353	0.738
0.03	500	0.60045	0.08788	0.387	1.149
0.04	500	0.72060	0.09823	0.458	1.122
0.05	500	0.82918	0.11849	0.517	1.245
0.06	500	0.93388	0.12474	0.619	1.388
0.07	500	1.05808	0.15011	0.708	1.682
0.08	500	1.18140	0.16706	0.787	1.794
0.09	500	1.27970	0.16783	0.841	1.897
0.1	500	1.39759	0.19249	0.952	2.167
Pooled	5000	0.88800	0.13000		
Bartlett's statistic		1507.06			
Degrees of freedom		9			
p-value		0			

Fig. 20: Computational time statistics when $pc = 0.80$ and pm varies from 0.01 to 0.1 with an interval of 0.01.



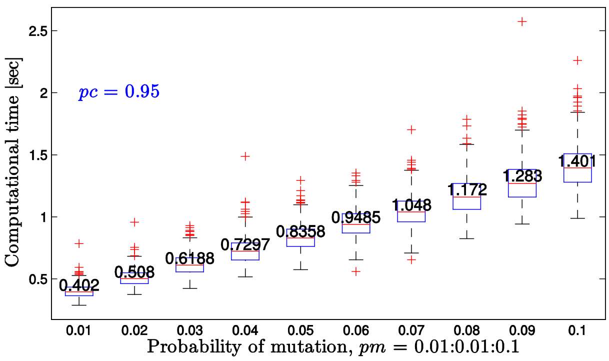
pm	runs	Mean[sec]	Std. Dev.	min [sec]	max [sec]
0.01	500	0.39068	0.04762	0.286	0.676
0.02	500	0.49806	0.06557	0.337	0.777
0.03	500	0.60814	0.08357	0.390	1.145
0.04	500	0.71697	0.10325	0.481	1.191
0.05	500	0.83525	0.11131	0.556	1.295
0.06	500	0.93652	0.13365	0.574	1.442
0.07	500	1.05184	0.14769	0.676	1.559
0.08	500	1.17318	0.16909	0.799	1.969
0.09	500	1.28131	0.18366	0.907	2.056
0.1	500	1.41074	0.18538	0.973	2.085
Pooled	5000	0.89027	0.13151		
Bartlett's statistic		1509.21			
Degrees of freedom		9			
p-value		0			

Fig. 21: Computational time statistics when $pc = 0.85$ and pm varies from 0.01 to 0.1 with an interval of 0.01.



pm	runs	Mean[sec]	Std. Dev.	min [sec]	max [sec]
0.01	500	0.39812	0.04868	0.288	0.763
0.02	500	0.50492	0.06910	0.357	0.761
0.03	500	0.60953	0.08540	0.418	0.972
0.04	500	0.71912	0.10093	0.480	1.132
0.05	500	0.83304	0.10740	0.536	1.235
0.06	500	0.93705	0.12843	0.610	1.454
0.07	500	1.05438	0.14616	0.704	1.925
0.08	500	1.17712	0.15364	0.822	1.762
0.09	500	1.27779	0.17547	0.881	1.923
0.1	500	1.38810	0.18558	0.998	2.529
Pooled	5000	0.88992	0.12761		
Bartlett's statistic		1361.55			
Degrees of freedom		9			
p-value		0			

Fig. 22: Computational time statistics when $pc = 0.90$ and pm varies from 0.01 to 0.1 with an interval of 0.01.



pm	runs	Mean[sec]	Std. Dev.	min [sec]	max [sec]
0.01	500	0.40200	0.05066	0.287	0.784
0.02	500	0.50800	0.06606	0.374	0.957
0.03	500	0.61884	0.08490	0.424	0.928
0.04	500	0.72966	0.10749	0.516	1.487
0.05	500	0.83582	0.11068	0.574	1.294
0.06	500	0.94852	0.11799	0.559	1.353
0.07	500	1.04761	0.13876	0.653	1.701
0.08	500	1.17235	0.15284	0.824	1.787
0.09	500	1.28251	0.17789	0.942	2.574
0.1	500	1.40142	0.18312	0.987	2.260
Pooled	5000	0.89467	0.12635		
Bartlett's statistic		1322.04			
Degrees of freedom		9			
p-value		0			

Fig. 23: Computational time statistics when $pc = 0.95$ and pm varies from 0.01 to 0.1 with an interval of 0.01.

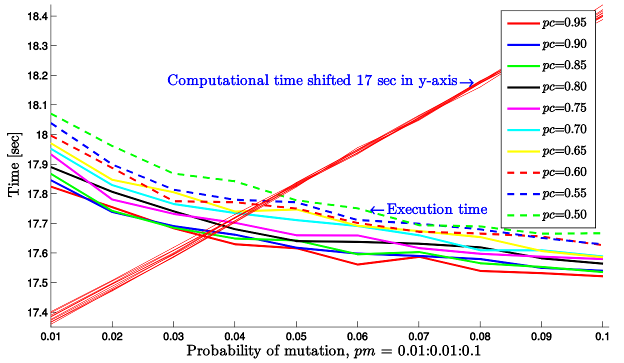


Fig. 24: Average Execution time and Computational time for all combinations between pc and pm.

approach as well as to show the improvement achieved by the proposed algorithm over the existing ones. The computational cost of the procedure itself is not a concern and is out of the scope of this work, as in industrial manipulators applications, the optimal trajectory planning is often done offline. In further work, it would be interesting to apply the presented parallel populations genetic algorithm procedure to trajectories with different interpolation functions, such as: fifth-order B-splines, harmonic, etc.

Appendix A

The values of a_{ij} and y_i in Eq. 7.

$$a_{11} = 3h_1 + 2h_2 + \frac{h_1^2}{h_2}, \quad a_{12} = h_2, \quad a_{21} = h_2 - \frac{h_1^2}{h_2},$$

$$a_{22} = 2(h_2 + h_3), \quad a_{23} = h_3, \quad a_{32} = h_3,$$

$$a_{33} = 2(h_3 + h_4), \quad a_{34} = h_4, \quad a_{n-3,n-4} = h_{n-3},$$

$$a_{n-3,n-3} = 2(h_{n-3} + hn - 2),$$

$$a_{n-3,n-2} = h_{n-2} - \frac{h_{n-1}^2}{h_{n-2}},$$

$$a_{n-2,n-3} = h_{n-2}$$

$$a_{n-2,n-2} = 3h_{n-1} + 2h_{n-2} + \frac{h_{n-1}^2}{h_{n-2}},$$

$$y_i = 6\left(\frac{q_{i+1} - q_i}{h_i} - \frac{q_i - q_{i-1}}{h_{i-1}}\right),$$

$$\begin{aligned} y_1 = {} & 6\left(\tfrac{q_3}{h_2} + \tfrac{q_1}{h_1}\right) \\ & -6\left(\tfrac{1}{h_1} + \tfrac{1}{h_2}\right)\left(q_1 + h_1\dot{q}_1 + \tfrac{h_1^2}{3}\ddot{q}_1\right) - h_1\ddot{q}_1, \end{aligned}$$

$$y_2 = \frac{6}{h_2}\left(q_1 + h_1\dot{q}_1 + \frac{h_1^2}{3}\ddot{q}_1\right) + \frac{6q_4}{h_3} - 6\left(\frac{1}{h_2} + \frac{1}{h_3}\right)q_3,$$

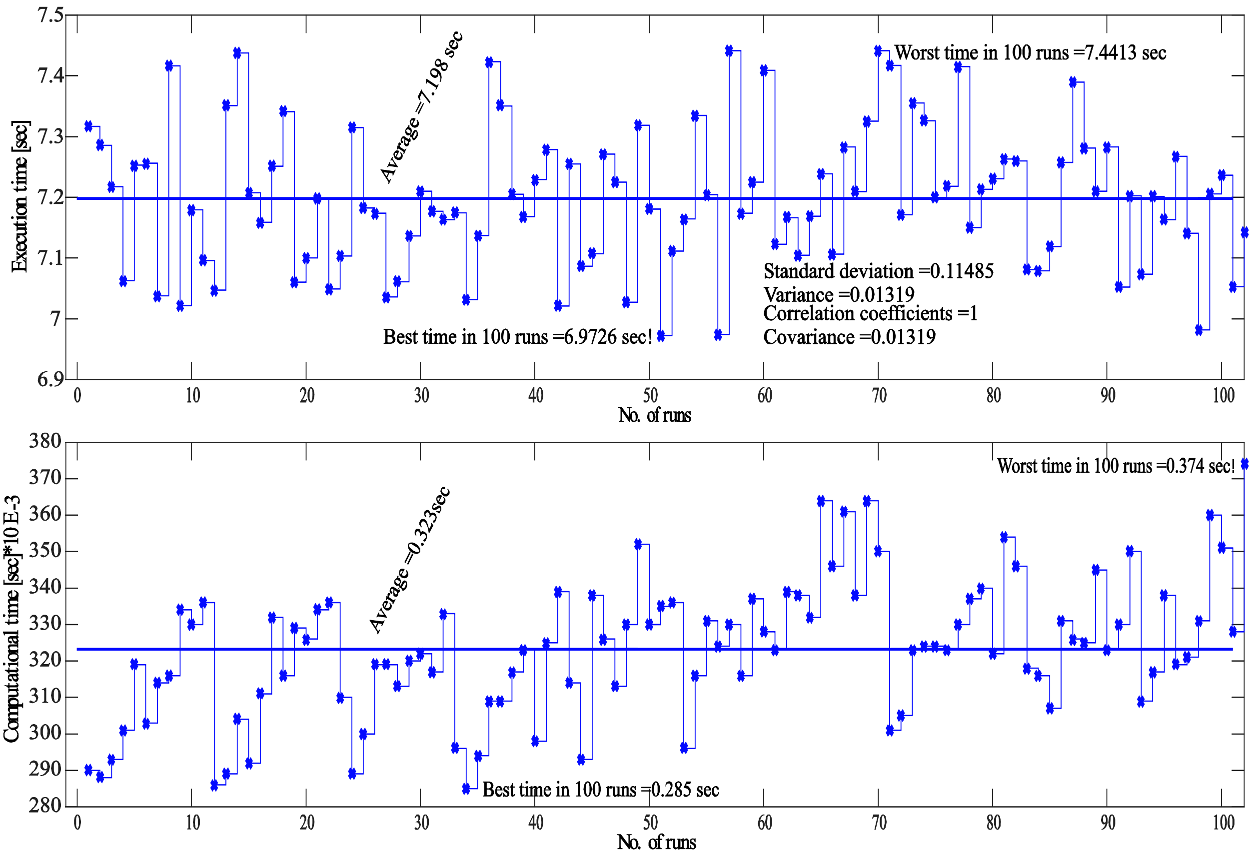


Fig. 25: Statistical analysis of the trajectory execution time and the computational cost of the PPGA1.

$$y_{n-3} = \frac{6}{h_{n-2}}\left(q_n - h_{n-1}\dot{q}_n + \frac{h_{n-1}^2}{3}\ddot{q}_n\right)$$
$$-6\left(\frac{1}{h_{n-2}} + \frac{1}{h_{n-3}}\right) q_{n-2} + \frac{6q_{n-3}}{h_{n-3}},$$

$$y_{n-2} = -6\left(\frac{1}{h_{n-1}} + \frac{1}{h_{n-2}}\right)\left(q_n - h_{n-1}\dot{q}_n + \frac{h_{n-1}^2}{3}\ddot{q}_n\right)$$
$$+6\left(\frac{q_n}{h_{n-1}} + \frac{q_{n-2}}{h_{n-2}}\right) - h_{n-1}\ddot{q}_n.$$

References

1. Abu-Dakka, F.J.: Trajectory planning for industrial robot using genetic algorithms. Ph.D. thesis, Universitat Politècnica de València (2011)
2. Abu-Dakka, F.J., Valero, F., Mata, V.: Evolutionary path planning algorithm for industrial robots. Advanced Robotics **26**(11-12), 1369–1392 (2012)
3. Ata, A.A., Myo, T.R.: Optimal point-to-point trajectory tracking of redundant manipulators using generalized pattern search. International Journal of Advanced Robotics Systems **2**(3) (2005)
4. Balkan, T.: A dynamic programming approach to optimal control of robotic manipulators. Mechanics Research Communications **25**(2), 225–230 (1998)
5. Bianco, C.G.L., Piazzi, A.: A semi-infinte optimization approach to optimal spline trajectory planning of mechanical manipulators. In: Semi-Infinite Programming, pp. 271–297. Springer (2001)
6. Bobrow, J.E., Dubowsky, S., Gibson, J.: Time-optimal control of robotic manipulators along specified paths. The international journal of robotics research **4**(3), 3–17 (1985)
7. Chettibi, T., Lehtihet, H., Haddad, M., Hanchi, S.: Minimum cost trajectory planning for industrial robots. European Journal of Mechanics-A/Solids **23**(4), 703–715 (2004)
8. Costantinescu, D., Croft, E.: Smooth and time-optimal trajectory planning for industrial manipulators along specified paths. Journal of Robotic Systems **17**(5), 233–249 (2000)
9. Craig, J.J.: Introduction to robotics: mechanics and control, vol. 3. Pearson Prentice Hall Upper Saddle River (2005)
10. Dong, J., Ferreira, P., Stori, J.: Feed-rate optimization with jerk constraints for generating minimum-time trajectories. International Journal of Machine Tools and Manufacture **47**(12), 1941–1955 (2007)
11. Dongmei, X., Daokui, Q., Fang, X.: Path-constrained timen-ioptimal robot control. In: Proceedings of IEEE International Conference on Robotics and Biomimetics, ROBIO'06, pp. 1095–1100. IEEE, Kunming (2006)
12. Elnagar, A., Hussein, A.: On optimal constrained trajectory planning in 3d environments. Robotics and Autonomous systems **33**(4), 195–206 (2000)
13. Fung, R.F., Cheng, Y.H.: Trajectory planning based on minimum absolute input energy for an lcd glass-handling robot. Applied Mathematical Modelling **38**(11), 2837–2847 (2014)

14. Gasparetto, A., Boscariol, P., Lanzutti, A., Vidoni, R.: Trajectory planning in robotics. Mathematics in Computer Science **6**(3), 269–279 (2012)
15. Gasparetto, A., Lanzutti, A., Vidoni, R., Zanotto, V.: Experimental validation and comparative analysis of optimal time-jerk algorithms for trajectory planning. Robotics and Computer-Integrated Manufacturing **28**(2), 164–181 (2012)
16. Gasparetto, A., Zanotto, V.: A new method for smooth trajectory planning of robot manipulators. Mechanism and Machine Theory **42**(4), 455–471 (2007)
17. Gasparetto, A., Zanotto, V.: A technique for time-jerk optimal planning of robot trajectories. Robotics and Computer-Integrated Manufacturing **24**(3), 415–426 (2008)
18. Huang, P., Chen, K., Yuan, J., Xu, Y.: Motion trajectory planning of space manipulator for joint jerk minimization. In: Proceedings of the International Conference on Mechatronics and Automation, ICMA2007, pp. 3543–3548. IEEE, Harbin (2007)
19. Izumi, T., Yokose, Y., Tamai, R.: Minimum energy path search for a manipulator in consideration of all nonlinear characteristics by ga and its experiments. Electrical Engineering in Japan **157**(3), 26–34 (2006)
20. Jamhour, E., André, P.: Planning smooth trajectories along parametric paths. Mathematics and Computers in Simulation **41**(5), 615–626 (1996)
21. Kahn, M.E., Roth, B.: The near-minimum-time control of open-loop articulated kinematic chains. Journal of Dynamic Systems, Measurement, and Control **93**(3), 164–172 (1971)
22. Kim, B.K., Shin, K.G.: Minimum-time path planning for robot arms and their dynamics. IEEE Transactions on Systems, Man and Cybernetics **SMC-15**(2), 213–223 (1985)
23. Kim, J., Kim, S.R., Kim, S.J., Kim, D.H.: A practical approach for minimum-time trajectory planning for industrial robots. Industrial Robot: An International Journal **37**(1), 51–61 (2010)
24. Kyriakopoulos, K.J., Saridis, G.N.: Minimum jerk path generation. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 364–369. IEEE, Philadelphia, PA (1988)
25. Lin, C.S., Chang, P.R., Luh, J.Y.S.: Formulation and optimization of cubic polynomial joint trajectories for industrial robots. IEEE Transactions on Automatic Control **28**(12), 1066–1074 (1983)
26. Lin, H.I.: A fast and unified method to find a minimum-jerk robot joint trajectory using particle swarm optimization. Journal of Intelligent & Robotic Systems **75**(3-4), 379–392 (2014)
27. Lo Bianco, C.G., Piazzi, A.: A hybrid algorithm for infinitely constrained optimization. International Journal of Systems Science **32**(1), 91–102 (2001)
28. Luh, J.Y., Lin, C.S.: Optimum path planning for mechanical manipulators. Journal of Dynamic Systems, Measurement, and Control **103**(2), 142–151 (1981)
29. Luo, L.P., Yuan, C., Yan, R.J., Yuan, Q., Wu, J., Shin, K.S., Han, C.S.: Trajectory planning for energy minimization of industry robotic manipulators using the lagrange interpolation method. International Journal of Precision Engineering and Manufacturing **16**(5), 911–917 (2015)
30. Perumaal, S., Jawahar, N.: Synchronized trigonometric s-curve trajectory for jerk-bounded time-optimal pick and place operation. International Journal of Robotics and Automation **27**(4), 385 (2012)
31. Piazzi, A., Visioli, A.: An interval algorithm for minimum-jerk trajectory planning of robot manipulators. In: Proceedings of the 36th IEEE Conference on Decision and Control, vol. 2, pp. 1924–1927. IEEE, San Diego, CA (1997)
32. Piazzi, A., Visioli, A.: Global minimum-time trajectory planning of mechanical manipulators using interval analysis. International journal of Control **71**(4), 631–652 (1998)
33. Piazzi, A., Visioli, A.: Global minimum-jerk trajectory planning of robot manipulators. IEEE Transactions on Industrial Electronics **47**(1), 140–149 (2000)
34. Richard, M.J., Dufour, F., Tarasiewicz, S.: Commande des robots manipulateurs par la programmation dynamique. Mechanism and Machine Theory **28**(3), 301–316 (1993)
35. Rubio, F., Valero, F., Lluís Sunyer, J., Garrido, A.: The simultaneous algorithm and the best interpolation function for trajectory planning. Industrial Robot: An International Journal **37**(5), 441–451 (2010)
36. Rubio, F.J., Valero, F.J., Suñer, J.L., Mata, V.: Simultaneous algorithm to solve the trajectory planning problem. Mechanism and Machine Theory **44**(10), 1910–1922 (2009)
37. Saramago, S., Steffen, V.: Optimization of the trajectory planning of robot manipulators taking into account the dynamics of the system. Mechanism and Machine Theory **33**(7), 883–894 (1998)
38. Saramago, S., Steffen Jr, V.: Trajectory modeling of robot manipulators in the presence of obstacles. Journal of Optimization Theory and Applications **110**(1), 17–34 (2001)
39. Saramago, S.F., Ceccarelli, M.: An optimum robot path planning with payload constraints. Robotica **20**(04), 395–404 (2002)
40. Saramago, S.F., Junior, V.S.: Optimal trajectory planning of robot manipulators in the presence of moving obstacles. Mechanism and Machine Theory **35**(8), 1079–1094 (2000)
41. Saramago, S.F., Steffen Júnior, V.: Dynamic optimization for the trajectory planning of robot manipulators in the presence of obstacles. Journal of the Brazilian Society of Mechanical Sciences **21**(3), 372–383 (1999)
42. Saravanan, R., Ramabalan, S.: Evolutionary minimum cost trajectory planning for industrial robots. Journal of Intelligent and Robotic Systems **52**(1), 45–77 (2008)
43. Saravanan, R., Ramabalan, S., Balamurugan, C.: Evolutionary optimal trajectory planning for industrial robot with payload constraints. The International Journal of Advanced Manufacturing Technology **38**(11-12), 1213–1226 (2008)
44. Saravanan, R., Ramabalan, S., Balamurugan, C., Subash, A.: Evolutionary trajectory planning for an industrial robot. International Journal of Automation and Computing **7**(2), 190–198 (2010)
45. Sato, A., Sato, O., Takahashi, N., Kono, M.: Trajectory for saving energy of a direct-drive manipulator in throwing motion. Artificial Life and Robotics **11**(1), 61–66 (2007)
46. Shin, K.G., McKay, N.D.: Minimum-time control of robotic manipulators with geometric path constraints. IEEE Transactions on Automatic Control **30**(6), 531–541 (1985)
47. Simon, D., Isik, C.: A trigonometric trajectory generator for robotic arms. International Journal of Control **57**(3), 505–517 (1993)
48. Tangpattanakul, P., Artrit, P.: Minimum-time trajectory of robot manipulator using harmony search algorithm. In:

Proceedings of the 6th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, ECTI-CON2009, vol. 1, pp. 354–357. IEEE, Pattaya, Chonburi (2009)
49. Thompson, S.E., Patel, R.V.: Formulation of joint trajectories for industrial robots using b-splines. IEEE Transactions on Industrial Electronics **IE-34**(2), 192–199 (1987)
50. Tse, K.M., Wang, C.H.: Evolutionary optimization of cubic polynomial joint trajectories for industrial robots. In: Proceedings of IEEE International Conference on Systems, Man, and Cybernetics, vol. 4, pp. 3272–3276. IEEE, San Diego, CA (1998)
51. Van Dijk, N., Van de Wouw, N., Nijmeijer, H., Pancras, W.: Path-constrained motion planning for robotics based on kinematic constraints. In: ASME 2007 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, pp. 1071–1080. American Society of Mechanical Engineers (2007)
52. Wall, M.: Galib: A c++ library of genetic algorithm components. Mechanical Engineering Department, Massachusetts Institute of Technology **87**, 54 (1996)
53. Wang, C.H., Horng, J.G.: Constrained minimum-time path planning for robot manipulators via virtual knots of the cubic b-spline functions. IEEE Transactions on Automatic Control **35**(5), 573–577 (1990)
54. Yazdani, M., Gamble, G., Henderson, G., Hecht-Nielsen, R.: A simple control policy for achieving minimum jerk trajectories. Neural Networks **27**, 74–80 (2012)