Robotica (2008) volume 26, pp. 753–765. © 2008 Cambridge University Press
doi:10.1017/S0263574708004359 Printed in the United Kingdom

Multiobjective trajectory planner for industrial robots with payload constraints

R. Saravanan†, S. Ramabalan*‡ and C. Balamurugan‡

†*Department of Mechatronics Engineering, Kumaraguru College of Technology, Coimbatore – Pin: 641 006, Tamil nadu*
‡*Faculty of CAD/CAM (P.G. Course), J. J. College of Engineering and Technology, Tiruchirapalli – Pin: 620 009*

(Received in Final Form: February 11, 2008. First published online: March 27, 2008)

SUMMARY
A general new methodology using evolutionary algorithms viz., Elitist Non-dominated Sorting Genetic Algorithm (NSGA-II) and Multi-objective Differential Evolution (MODE), for obtaining optimal trajectory planning of an industrial robot manipulator (PUMA 560 robot) in the presence of fixed and moving obstacles with payload constraint is presented. The problem has a multi-criterion character in which six objective functions, 32 constraints and 288 variables are considered. A cubic NURBS curve is used to define the trajectory. The average fuzzy membership function method is used to select the best optimal solution from Pareto optimal fronts. Two multi-objective performance measures namely solution spread measure and ratio of non-dominated individuals are used to evaluate the strength of Pareto optimal fronts. Two more multi-objective performance measures namely optimiser overhead and algorithm effort are used to find computational effort of the NSGA-II and MODE algorithms. The Pareto optimal fronts and results obtained from various techniques are compared and analysed. Both NSGA-II and MODE are best for this problem.

KEYWORDS: Optimal trajectory planning; NURBS; Payload constraints; Evolutionary algorithms—Elitist non-dominated sorting genetic algorithm (NSGA-II), Multi-objective differential evolution (MODE).

1. Introduction

In the field of robotics, the development of robots that accept high-level descriptions of tasks and execute these tasks without intervention from their environment is very much desired. The task of an autonomous robot is to carry payload, as it moves on its way from one placement to another specified placement. The capability of robots to complete tasks or entire missions autonomously relies heavily on their ability to plan. The problem of finding a path for the above is referred to as motion-planning problem. Most of today's operational robots are not fully autonomous and they need to deal with certain instances of the motion-planning problem during their operation. Optimal motion planning for robot manipulators is difficult because of the highly non-linear and coupled nature of manipulator dynamics, together with the complex system constraints, which include limitation on actuator forces/torques and the payload constraints.

In order to maximize the speed of operation, which affects the productivity in industrial situations, it is necessary to minimize the total travelling time of robot. Many research works have been carried out to get minimum time trajectories.[1–5] Planning the robot trajectory by using energetic criteria provides several advantages: (1) It yields smooth trajectories easy to track and reduce the stresses to the actuators and to the manipulator structure and (2) It saves energy, which is desirable in several applications, such as those with a limited capacity of energy source (e.g., robots for spatial or submarine exploration). Examples of energy optimal trajectory planning are provided in some literatures. Both optimal travelling time and the minimum mechanical energy of the actuators are considered together as objective functions in some literatures.[1–6] Saramago *et al.*[5] used a method based on the sequential optimization technique (SOP) to do path planning for the PUMA 560 robot in consideration of payload constraints. They used B-spline curve to represent the trajectory. Fields of research such as computer graphics, geometric design, and robotics (motion planning) prefer smooth trajectories that are achieved by minimization of the joint jerks[7,8] and accelerations.[9] The positive effects induced by minimization of the joint jerks and accelerations are (1) Errors during trajectory tracking are reduced, (2) Stresses to the actuators and structure of the manipulator are reduced, (3) Excitation of resonance frequencies of the robot is limited, and (4) A co-ordinated and natural robot motion is yielded. A measure of manipulability is very useful in manipulator designing, task planning, and enables manipulators recover faster from the escapable singular points. To obtain a practical trajectory (robot does not loose any degree of freedom at any stage), the manipulability measure can be used as decision criteria for robot trajectory planning.[10,11] The industrial robots, especially material handling robots have to carry variety of products with different weights (payloads). The trajectory followed by a robot manipulator for doing any action is much dependent on payload.[5] So it is mandatory to consider the payload constraints in the trajectory planning for industrial robots. In any industrial situation, the robot will have several obstacles in its working space. So we must consider the

* Corresponding author. E-mail: cadsrb@gmail.com

obstacle avoidance criteria while doing an optimal trajectory planning.

In order to get the above benefits, we need to consider all the objective functions in a combined manner while doing optimal trajectory planning. None of literatures by far available had considered all these objective functions in a combined manner.

Tatematsu *et al.*[12] presented an approach to track the moving target with mobile robot. The task is difficult because the mobile robot has nonholonomic constraint. Considering the motion constraint of mobile robot, planning of the trajectory that the robot can follow is indispensable for wheeled type mobile robot control. NURBS curve is used for trajectory planning. The continuity to second derivative and convex hull, which are the characteristics of the NURBS curve that is best for trajectory planning of the mobile robot. By this method, higher efficient tracking is realized. Simulations confirm the validity of the method.

In this paper, cubic NURBS functions are used for constructing joint trajectories since they have the following advantages:

(a) They represent free form shape remarkably with a little data and can be well defined in the mathematic form.
(b) They allow local controllability, which implies the local changes in shape are confined to the NURBS parameters local to that change.
(c) They give ability to control smoothness and curvature continuity.
(d) They possess the characteristic of shape invariance under affine transformation, which means, the affine transformed curve is still a NURBS curve whose control points and weights are related to the original curve control points and weights through this transformation.

The choice of NURBS as a shape descriptor, not only offers a common mathematical form for representing free-form shapes but also geometric shapes. The difference between NURBS and B-spline is that NURBS includes a non-uniform control point vector and an additional parameter, which is weight. Inclusion of weight as an additional parameter adds an extra degree of freedom to NURBS and facilitates the representation of a wide variety of shapes. Furthermore, the use of non-uniform control point vectors allows better shape control and the modelling of a much larger class of shapes than the uniform knot vector used in B-spline. With these additional parameters, NURBS allows a higher compact representation, which effectively reduces the original number of the boundary points required to represent the robot trajectory. Therefore, it is strongly believed that NURBS is a better and accurate trajectory shape descriptor.

The methods that are used in the literatures such as sequential unconstrained minimization technique (SUMT),[1–5] interval analysis,[7] sequential quadratic programming (SQP)[6,8] and numerical iterative procedure[9] to tackle the complex instances (obstacles environment) have some notable drawbacks viz., (1) they may fail to find optimal path (or spend a lot of time and memory storage to find one), (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. To overcome the above drawbacks, the evolutionary algorithms can be used.[11,13] The advantages of evolutionary techniques are (1). They are population-based search, so global optimal solution is possible. (2) They do not need any auxiliary information like gradients, derivatives, etc. (3) They can solve complex and multimodal problems for global optimality. (4) They are problem independent, i.e., suitable for solving all types of problems.

The major limitations of the previous works[1–10] are

- They have only used conventional optimisation techniques such as SUMT,[1–5] SQP[6,8] and interval analysis.[7]
- None of literatures considered all the necessary criteria (All important objective functions e.g., Minimization of travelling time, energy, joint jerks, joint accelerations, and maximization of manipulability measure, payload constraint, obstacle avoidance constraint, the actuator limit constraints, and NURBS curve to define the robot joint trajectory) in a combined manner.

To overcome above drawbacks, in this paper two evolutionary optimisation techniques NSGA-II and MODE are proposed to do optimal trajectory planning for PUMA 560 robot manipulator by considering all objective functions (Minimization of travelling time, energy, joint jerks, joint accelerations and maximization of manipulability measure), payload constraint, obstacle avoidance constraint and the actuator limit constraints. Also cubic NURBS curves are used to define the robot joint trajectories. The average fuzzy membership function method is used to select the best optimal solution from Pareto optimal fronts (optimal solution trade-offs). Two multiobjective performance measures namely solution spread measure and ratio of non-dominated individuals are used to evaluate the strength of Pareto optimal fronts. Two more multiobjective performance measures namely optimiser overhead and algorithm effort are used to find the computational effort of the NSGA-II and MODE algorithms.

The proposed optimisation methods have following advantages: (1) global optimal solution is possible, (2) they consider all the important decision criteria for trajectory planning of industrial robot manipulators, (3) they are easy to program and implement, (4) they ensure that the resulting optimised trajectory is a smoother, faster, safer, non-singular, and need minimum actuators effort and power, (5) they can also be extended to get optimised trajectory of other types of robots, (6) they consider both kinematic and dynamic aspects of the robot, (7) they consider both payload and obstacle avoidance constraints, (8) they are computationally superior and faster, (9) they offer Pareto optimal fronts that will offer more number of optimal solutions for user's choice, (10) they use fuzzy average membership function method to find best optimal solution from the Pareto optimal fronts and (11) They use NURBS curve to define the trajectory.

This paper is organized as follows. The problem formulation is presented in Section 2. In Section 3, the proposed NSGA-II and MODE techniques are presented to obtain the optimal solutions. Section 4 deals a method and four multi-objective performance metrics used for evaluating the

https://doi.org/10.1017/S0263574708004359 Published online by Cambridge University Press

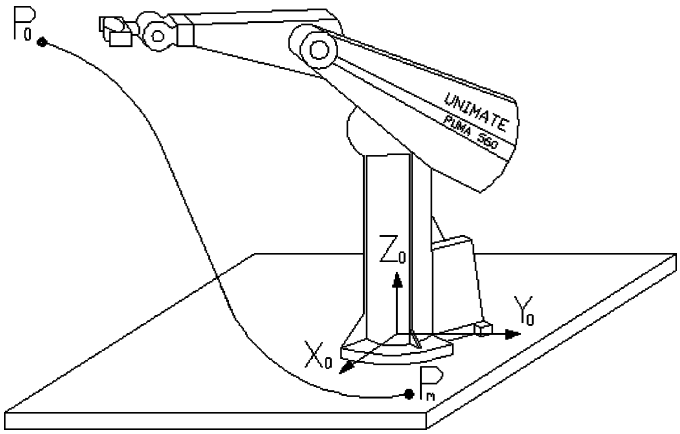

Fig. 1. A PUMA 560 robot manipulator and a prescribed path given by initial and final points.

proposed algorithms. In Section 5, a numerical example of the industrial robot with 6 degrees of freedom (PUMA 560 robot) is presented to illustrate the use of the proposed NSGA-II and MODE techniques to find out the optimal solutions. In Section 6, the results obtained in various methods are presented and compared. The conclusions are presented in Section 7.

2. Problem Formulation

The industrial Robot manipulator with 6 degrees of freedom (dof) i.e., PUMA 560 robot (Fig. 1) is considered. The target is to move the robot that carries a payload from an initial configuration to a final configuration in its workspace while optimising several objective functions of the robot considering the robot physical constraints, payload constraints and actuator limits. The problem has six objective functions, 32 constraints and 288 variables.

Minimization of travelling time, total energy involved in the motion, joint jerks, joint accelerations, penalty for obstacle avoidance and maximization of singularity avoidance are considered as objective functions. The singularity avoidance is chosen in the form of the manipulability measure. Maximizing the manipulability measure will force the manipulator away from the singularity. If the penalty function to guarantee free collision-motion is zero, it indicates that there is no collision between the robot and the obstacles.

The multicriterion optimisation problem is defined as follows:

Minimize

Total travelling time between initial and final configurations = z1 = T

Total energy involved in the motion = $z2 = \int_0^T \sum_{i=1}^n (u_i(t)\,\dot{q})^2 \,.\, dt$

Penalty for obstacle avoidance = $z3 = f_{dis}$

Integral of squared link jerks = $z4 = \int_0^T \sum_{i=1}^n (\dddot{q}_i^{\,2})\, dt$

Integral of squared link accelerations = $z5 = \int_0^T \sum_{i=1}^n (\ddot{q}_i)\, dt$

Maximize

Manipulability measure = z6 = |det(J)| (1a–f)

Subject to,

1. Displacement constraint.

$$|q_{ji}(t)| \le q_{ji}^{\max} \tag{2}$$

2. Velocity constraint.

$$|\dot{q}_{ji}(t)| \le \dot{q}_{ji}^{\max} \tag{3}$$

3. Acceleration constraint.

$$|\ddot{q}_{ji}(t)| \le \ddot{q}_{ji}^{\max} \tag{4}$$

4. Jerk constraint.

$$|\dddot{q}_{ji}(t)| \le \dddot{q}_{ji}^{\max} \tag{5}$$

5. Force/torque constraint.

$$|u_{ji}(t)| \le u_{ji}^{\max} \quad \text{for j} = 1, 2, \ldots, \text{n and}$$
$$\text{i} = 1, 2, \ldots\ldots, \text{m} - 1 \tag{6}$$

6. Payload constraint.

$$Fg_{\min} \le F_k \le Fg_{\max} \tag{7}$$

where, u_{ji} is the generalized forces (torques), J is Jacobian matrix of the robot, n represents the robot joints and 'm' represents the knots used to construct the trajectories, F_k is the grasping forces of the fingers (F_1 or F_2), Fg_{min} and $Fg_{\max}$ are minimum and maximum grasping forces of the fingers. q_{ij} is robot joint displacement, $\dot{q}_{ij}$ is robot joint velocity, $\ddot{q}_{ij}$ is robot joint acceleration, $\dddot{q}_{ij}$ is robot joint jerk, $q_{ij}^{\max}$ is the maximum value of the robot joint displacement, $\dot{q}_{ij}^{\max}$ is the maximum value of the robot joint velocity, $\ddot{q}_{ij}^{\max}$ is the maximum value of the robot joint acceleration, $\dddot{q}_{ij}^{\max}$ is the maximum value of the robot joint jerk and $u_{ij}^{\max}$ is the maximum value of the robot joint torque.

2.1. Kinematic and dynamic models

According to Saramago *et al.*[5] the generalized forces are calculated as:

$$u_i = \sum_{j=1}^{n} D_{ij}\,\ddot{q}_k + \sum_{i=1}^{n}\sum_{k=1}^{i} C_{ijk}\,\dot{q}_k\,\dot{q}_m + G_i \tag{8}$$

$$D_{ij} = \sum_{k=1}^{j} Tr[U_{jk} J_i (U_{ji})^T] \tag{9}$$

$$C_{ijk} = \sum_{m=1}^{j} Tr[U_{jkm} J_i (U_{ji})^T] \tag{10}$$

$$G_i = \sum_{j=1}^{n} -m_j g^T (U_{ji} \overline{r_j}) \tag{11}$$

where, D_{ij} is the inertial system matrix, C_{ijk} is the coriolis and centripetal forces matrix, and G_i is the gravity-loading vector. J_i is the moments of Inertia, $\overline{r_j}$ is centre of mass,

https://doi.org/10.1017/S0263574708004359 Published online by Cambridge University Press

and g is acceleration due to gravity with respect to the base coordinate system.

The following equation gives energy dissipation, which considers both friction (coulomb) and linear viscous damping.

$$F_{diss} = f_c\, sign(\dot{q}) + f_d\dot{q} \tag{12}$$

where, f_c is the Coulomb force coefficient and f_d is the viscous damping coefficient.

2.2. Trajectory representation

Only the initial and final points to construct the joint trajectories are given. A cubic NURBS curve is used to define the trajectory.

NURBS curves have various useful properties such as allowing smoothness and possibility of local modifications, which facilitate the representation of robot trajectories. These properties of NURBS curves are ideal for the design of complicated geometry, making them a standard tool in computer-aided design and manufacturing (CAD/CAM) and graphics fields.

NURBS are represented parametrically by the following equation:[17]

$$P(u) = \frac{\sum_{i=0}^{n} N_{i,k}(u) W_i V_i}{\sum_{i=0}^{n} N_{i,k}(u) W_i} = \sum_{i=0}^{n} V_i R_{i,k}(u) \tag{13}$$

and

$$R_{i,k}(u) = \frac{N_{i,k}(u) W_i}{\sum_{i=0}^{n} N_{i,k}(u) W_i} \tag{14}$$

where, V_i is the control point, W_i is its weighting factor, $n+1$ is the number of control points, and k is the order of the NURBS. $N_{i,k}(u)$ and $R_{i,k}(u)$are called the kth order basis function and rational basis function. Recursive formulae for the blending function $N_{i,k}(u)$ can be found as

$$N_{i,1}(u) = \begin{cases} 1 & for \quad u_i \le u < u_{i+1} \\ 0 & otherwise \end{cases} \tag{15}$$

$$N_{i,k}(u) = \frac{u - u_i}{u_{i+k-1} - u_i} N_{i,k-1}(u) + \frac{u_{i+k} - u}{u_{i+k} - u_{i+1}} N_{i+1,k-1}(u) \tag{16}$$

where, $U = [u_i, \ldots, u_{i+k}]$ represents the knot vector. Since the m^{th} derivative of $P(u)$ is given by

$$P^{(m)}(u) = \sum_{i=0}^{n} V_i R_{i,k}^{(m)}(u) \tag{17}$$

To find velocity, acceleration and jerk of the robot joint, we need first, second, and third derivatives of the single rational B-spline $R_{i,k}(u)$as follows:

$$R_{i,k}^{(1)}(u) = \frac{W_i N_{i,k}^{(1)}(u)}{\sum_{i=0}^{n} W_i N_{i,k}(u)} - \frac{W_i N_{i,k}(u) \sum_{i=0}^{n} W_i N_{i,k}^{(1)}(u)}{\left[\sum_{i=0}^{n} W_i N_{i,k}(u)\right]^2} \tag{18}$$

$$R_{i,k}^{(2)}(u) = \frac{W_i N_{i,k}^{(2)}(u)}{\sum_{i=0}^{n} W_i N_{i,k}(u)} - \frac{\left[2W_i N_{i,k}^{(1)}(u) \sum_{i=0}^{n} W_i N_{(i,k)}^{(1)}(u) + W_i N_{i,k}(u) \sum_{i=0}^{n} W_i N_{(i,k)}^{(2)}(u)\right]}{\left[\sum_{i=0}^{n} W_i N_{i,k}(u)\right]^2} + \frac{2W_i N_{i,k}(u) \left[\sum_{i=0}^{n} W_i N_{i,k}(1)(u)\right]^2}{\left[\sum_{i=0}^{n} W_i N_{i,k}(u)\right]^3} \tag{19}$$

Similarly jerk of the robot joint can be found from third derivatives of the single rational B-spline $R_{i,k}(u)$, where, $N_{i,k}^{(m)}(u)$ is the mth derivative of the basis function. The general formula for computing $N_{i,k}^{(m)}(u)$ is given by

$$N_{i,k}^{(m)}(u) = (k-1)\left[\frac{N_{i,k-1}^{(m-1)}(u)}{u_{i+k-1} - u_i} - \frac{N_{i+1,k-1}^{(m-1)}(u)}{u_{i+k} - u_{i+1}}\right] \tag{20}$$

where, $P(u)$ is the vector to the point defined at some value of u, V_i is the control points (in the 3D case, $Vi = \{X_i, Y_i, Z_i\}^T$), W_i is the weight factor, and $n$is the number of control points.

The trajectory has to pass through three intermediate points. So there are four segments in between initial and final robot configurations. Each segment is defined by a third-order NURBS curve with four control points as shown in Fig. 2. The weight vector for control points in each segment is $W_i = \{1, 2, 2, 1\}$. Increasing (*resp.*, decreasing) the value of weight W_i pulls (*resp.*, pushes) the curve toward (*resp.*, away from) control point V_i. When the value of W_i becomes infinity, the

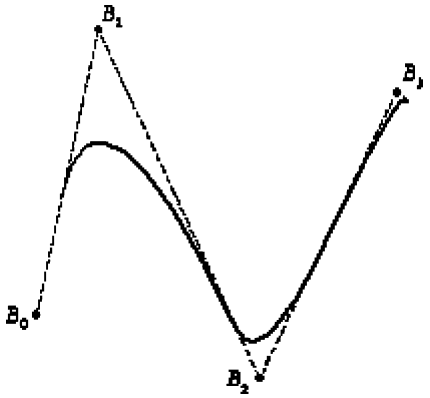

Fig. 2. Trajectory of the end effector.

https://doi.org/10.1017/S0263574708004359 Published online by Cambridge University Press

curve passes through control point V_i and when W_i is zero, control point V_i has no impact on the curve. There will be significant effect of number of control points on the results. If we increase number of control points, we may get more accurate and best optimal solution. But it will increase the computational time and effort. So in this paper, number of control points is limited to four in each segment. In future work of this research paper, Effect of number of control points on the results will be analysed.

In the optimisation of the trajectories the decision variables are the NURBS control points V_i.

2.3. Obstacle avoidance

The distance between potentially colliding parts is expressed as obstacle avoidance. Further the motion is represented by using translation and rotational matrices. When obstacles are found in the workspace it is necessary to add a penalty function in the multicriterion optimisation problem to guarantee free-collision motion. The idea is to circumscribe each obstacle into a specific sphere. Let (X_0, Y_0, Z_0) be the centre of an obstacle and r_0 be the radius of the sphere that circumscribe this obstacle. The trajectory points (X, Y, Z), which are located outside the sphere, are accepted according to the equation.

$$r_t = \sqrt{(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2} > r_0 \qquad (21)$$

where, r_t is the distance between the centre of the obstacle and a trajectory points as represented in Fig. 3. If Eq. (21) is verified, the trajectory is out of the sphere and the penalty function (f_{dis}) is zero. If the trajectory is tangent or crossing the sphere the multicriterion cost function will be penalized:

$$r_t > r_0 \Rightarrow f_{dis} = 0$$

If

$$r_t \leq r_0 \Rightarrow f_{dis} = \sum_{i=1}^{nobs} \frac{1}{(\min r_e)^2} \qquad (22)$$

where, nobs is the total number of obstacles in the workspace.

There are situations according to the topology of the obstacle. Where it is more likely to circumscribe the obstacle by an ellipsoid as shown in Fig. 4.

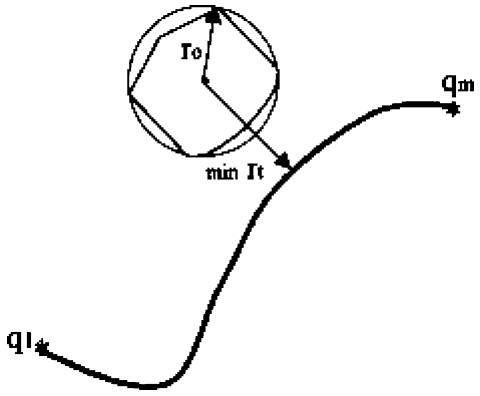

Fig. 3. Obstacle circumscribed by a sphere.

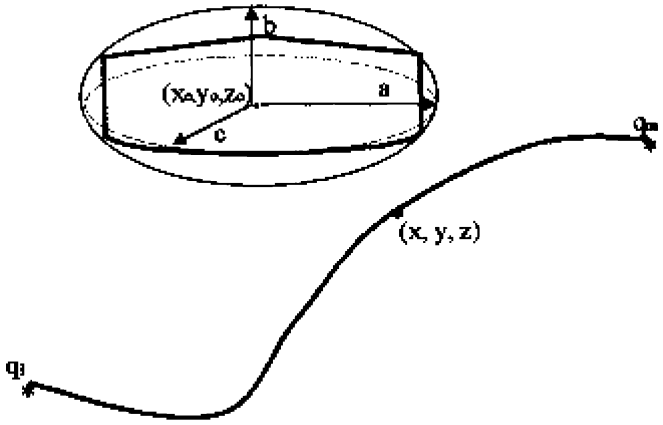

Fig. 4. Obstacle circumscribed by an ellipsoid.

Let a, b, c be the semi-axes of the circumscribing ellipsoids. Applying the same principle used for the circumscribing spheres. The trajectory points, which are located outside the ellipsoid, are accepted according to the equation:

$$r_e = \frac{(x - x_0)^2}{a^2} + \frac{(y - y_0)^2}{b^2} + \frac{(z - z_0)^2}{c^2} \qquad (23)$$

where r_e is the eccentricity.

Penalization is used in this case as below:

$$r_t > 1 \Rightarrow f_{dis} = 0$$

If

$$r_t \leq 1 \Rightarrow f_{dis} = \sum_{i=1}^{nobs} \frac{1}{(\min r_e)^2} \qquad (24)$$

This way the optimal control problem is to optimise the multicriterion functions defined by Eq. (1a–f) using the penalty function given by Eq. (22) or Eq. (24) taking into account the kinematic, dynamic, and obstacle avoidance constraints.

Obstacle avoidance is obtained by adding penalty functions to the multicriterion optimisation problem. Besides, constraints that describe minimal acceptable distance between potentially colliding parts are also included to the general non-linear optimisation problem. The obstacles are protected by spherical or hiper-spherical security zones, which are never penetrated by the end-effector.

2.4. Formulation of the grasping forces in the gripper

Usually, industrial robots use two-finger grippers for grasping purpose. So a two-finger gripper is considered here to grasp the object (Fig. 5). Grasping can be defined as the capability of a mechanical end-effector to establish a contact between its fingers and object. Grasp configurations are achieved so that a static equilibrium exists between the grasping forces by the fingers on the object. The grasping forces can be determined based on the characteristics of the object, such as its weight and shape. However, in most manipulator tasks inertia forces are also considered depending on the specifications of the motion trajectories.[5]

https://doi.org/10.1017/S0263574708004359 Published online by Cambridge University Press

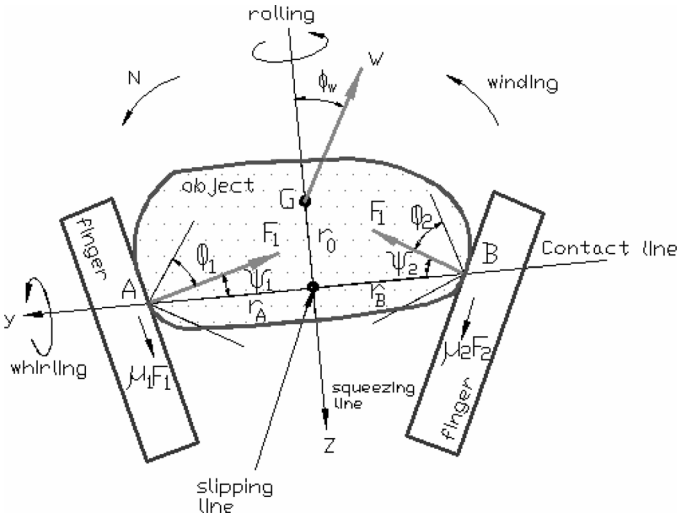

Fig. 5. Configuration and forces for a planar grasp with two fingers.

This paper uses an approach for computing the grasping forces as a function of the inertia forces too.

Let F_1 and F_2 be the grasping forces, which are exerted by the fingers. Usually they are not equal, since the contact points A and B are not generally located in same relative position on the two fingers. Similarly, friction can be evaluated at the points A and B through the coefficients μ_1 and μ_2. In Fig. 5 the grasping configuration of an object with respect to the fingers gives the angles ψ_1 and ψ_2, which strongly depend on the orientation of the fingers, position of the contact points, and shape of the object and fingers. These angles can differ from each other similarly to F_1 and F_2. r_A and r_B represents the distances of A and B, respectively from the squeezing line. $W = m_{object}\, g$ is the weight vector of the object and it is oriented with an angle ψ_w with respect to the perpendicular axis to the plane yz and G_{xyz} is a suitable frame fixed on the grasped object as shown in Fig. 5. N is external torque acting on the object and it includes the inertial actions due to the manipulator movement.

The static equilibrium of a grasped object can be expressed along the directions of the contact and squeezing lines, as outlined by Saramago *et al.*[5] in term of forces as

$$\begin{aligned} &F_1 \cos\psi_1 - F_2 \cos\psi_2 + \mu_1 F_1 \sin\psi_1 - \mu_2 F_2 \sin\psi_2 \\ &\quad + m_{object}\,(g \cos\psi_w \sin\Phi_w + a_y) = 0 \\ &-F_1 \sin\psi_1 - F_2 \sin\psi_2 + \mu_1 F_1 \cos\psi_1 + \mu_2 F_2 \cos\psi_2 \\ &\quad + m_{object}(g \cos\psi_w \cos\Phi_w + a_z) = 0 \end{aligned} \tag{25}$$

and in term of torque as

$$\begin{aligned} &r_A F_1(\sin\psi_1 - \mu_1 \cos\psi_1) - r_B F_2({}_1\sin\psi_2 - \mu_2 \cos\psi_2) \\ &\quad - N - r_G m_{object}(g + a_y)\sin\Phi_w = 0 \end{aligned} \tag{26}$$

where the acceleration components a_y and a_z of the centre point on the manipulator extremity can be computed as

$$\begin{bmatrix} a_x \\ a_y \\ a_z \\ 1 \end{bmatrix} = \left(\sum_{j=1}^{i} \left(\frac{\partial T_0^i}{\partial q_j} \ddot{q_j} \right) + \sum_{j=1}^{i} \left(\frac{\partial^2 T_0^i}{\partial q_j \partial q_k} \dot{q_j}\, \dot{q_k} \right) \right) \begin{bmatrix} r_{Gx} \\ r_{Gy} \\ r_{Gz} \\ 1 \end{bmatrix} \tag{27}$$

The distance of the gravity centre of the grasped object is indicated though vector r_G with components r_{Gx}, r_{Gy}, r_{Gz} and T_0^i is the homogeneous transformation matrix. The corresponding velocity components of the gravity centre point can be calculated as

$$\begin{bmatrix} v_x \\ v_y \\ v_z \\ 1 \end{bmatrix} = \left(\sum_{j=1}^{i} \left(\frac{\partial T_0^i}{\partial q_j} \dot{q_j} \right) \right) \begin{bmatrix} r_{Gx} \\ r_{Gy} \\ r_{Gz} \\ 1 \end{bmatrix} \tag{28}$$

The payload constraints are expressed in terms of feasible range of grasping forces F1 and F2.

3. Proposed Methods

To show the superior nature of evolutionary algorithms, this paper proposes two evolutionary optimisation techniques such as Elitist Non-dominated Sorting Genetic Algorithm (NSGA-II) and Multi Objective Differential Evolution (MODE).

3.1. Elitist non-dominated sorting genetic algorithm (NSGA-II)

Deb *et al.*[14] proposed the NSGA-II algorithm. Essentially, NSGA-II differs from non-dominated sorting Genetic Algorithm (NSGA) implementation in a number of ways. Firstly, NSGA-II uses an elite-preserving mechanism, thereby assuring preservation of previously found good solutions. Secondly, NSGA-II uses a fast non-dominated sorting procedure. Thirdly, NSGA-II does not require any tuneable parameter, thereby making the algorithm independent of the user.

Initially, a random parent population P_o created. The population is sorted based on the non-domination. A special book-keeping procedure is used in order to reduce the computational complexity down to $O(N^2)$. Each solution is assigned a fitness equal to its non-domination level (1 is the best level). Thus, minimization of fitness is assumed. Binary tournament selection, recombination, and mutation operators are used to create a child population Q_o of size N. Thereafter, we use the following algorithm in every generation. First, a combined population $R_i = P_i U Q_i$ is formed. This allows parent solutions to be compared with the child population, thereby ensuring elitism. The population R_i is of size 2N. Then, the population R_i is sorted according to non-domination. The new parent population P_{i+1} is formed by adding solutions from the first front and continuing to other fronts successively till the size exceeds N. Thereafter,

https://doi.org/10.1017/S0263574708004359 Published online by Cambridge University Press

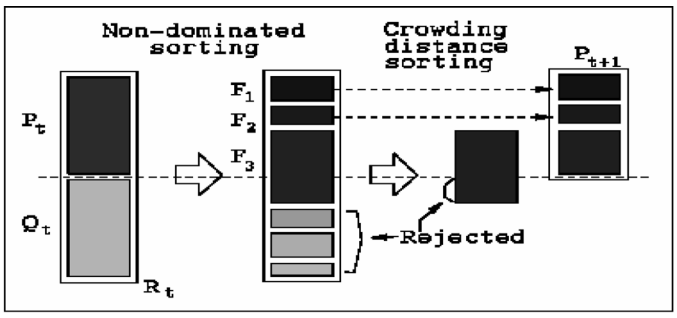

Fig. 6. An iteration of the NSGA-II procedure.

the solutions of the last accepted front are sorted according to a crowded comparison criterion and the first N points are picked. Since the diversity among the solutions is important, we use a partial order relation $\geq_n$ as follows:

$$i \geq_n j \quad \text{if}(i_{rank} < j_{rank}) \text{ or } ((i_{rank} = j_{rank}) \text{ and } (i_{fitness} > j_{fitness})$$

That is, between two solutions with differing nondomination ranks we prefer the point with the lower rank. Otherwise, if both the points belong to the same front then we prefer the point, which is located in a region with lesser number of points (or with larger crowded distance). This way solutions from less dense regions in the search space are given importance in deciding which solutions to choose from R_i. This constructs the population P_{i+1}. This population of size N is now used for selection, crossover and mutation to create a new population Q_{i+1} of size N. We use a binary tournament selection operator but the selection criterion is now based on the crowded comparison operator $\geq_n$. The above procedure is continued for a specified number of generations.

It is clear from the above description that NSGA-II uses (i) a faster non-dominated sorting approach, (ii) an elitist strategy, and (ii) no niching parameter. Diversity is preserved by the use of crowded comparison criterion in the tournament selection and in the phase of population reduction. NSGA-II has been shown to outperform other current elitist multi-objective Evolutionary Algorithms on a number of difficult test problems. Fig. 6 shows an iteration of the proposed NSGA-II procedure.

The values of the parameter that have been used in the NSGA-II technique are:

Variable type = Real variable, Population size = 100, Crossover probability = 0.6, Real-parameter mutation probability = 1, Real-parameter SBX parameter = 10, Real-parameter Mutation parameter = 100, Total number of generations = 100.

3.2. Multi-objective differential evolution (MODE)

Multi-Objective Differential Evolution (MODE)[15] can be categorized into a class of floating-point encoded evolutionary algorithms. The theoretical framework of MODE is very simple and MODE is computationally inexpensive in terms of memory requirements and CPU times. Thus, nowadays MODE has gained much attention and wide application in a variety of fields. Among the MODE's advantages are its simple structure, ease of use, speed, and robustness.

In a multi-objective domain, the goal is to identify the Pareto optimal solution set. In this proposed multi-objective differential evolution (MODE), a Pareto-based approach is introduced to implement the selection of the best individuals. Firstly, a population of size, NP, is generated randomly and the fitness functions are evaluated. At a given generation of the evolutionary search, the population is sorted into several ranks based on dominance concept. Secondly, Differential Evolution (DE) operations are carried out over the individuals of the population. The fitness functions of the trial vectors, thus formed, are evaluated. One of the major differences between DE[16] and MODE is that the trial vectors are not compared with the corresponding parent vectors. Instead, both the parent vectors and the trial vectors are combined to form a global population of size, 2*NP. Then, the ranking of the global population is carried out followed by the crowding distance calculation. The best NP individuals are selected based on its ranking and crowding distance. These act as the parent vectors for the next generation. The procedure is carried out until the entire selected best NP individuals have a rank of one. The Pseudo code for MODE algorithm is presented in Appendix. The values of the parameter that have been used in the proposed MODE technique are

Strategy = MODE/rand/1/bin, crossover constant CR = 0.9, population size NP = 500, F = 0.5 and total number of generations = 100.

4. Performance Measures and Methods for Multi-Objective Optimisation

In this section, one method and four performance metrics are recommended and applied to examine the strength and weaknesses of the proposed multi-objective evolutionary algorithms. The average fuzzy membership function method is used to select best optimal solution from Pareto optimal fronts. Two multiobjective performance measures namely solution spread measure and ratio of non-dominated individuals are used to evaluate the strength of Pareto optimal fronts. Two more multiobjective performance measures namely optimiser overhead and algorithm effort are used to find computational effort of the NSGA-II and MODE algorithms. These methods and metrics are chosen since they have been widely used for performance comparisons in multi-objective optimisation.

4.1. Fuzzy average membership function (μ_{avg}) method

The deterministic models proposed in the literature suffer from the limitation in a real world optimal trajectory planning due to the fact that a decision maker does not have sufficient information related to the different criteria. These data are typically fuzzy in nature. All the above-referred deterministic methods lack the capability to handle the linguistic vagueness of fuzzy type. The optimal results obtained from these deterministic formulations may not serve the real purpose of modelling the problem. A consideration to incorporate information vagueness (Fuzziness) in the real world optimal trajectory planning has not been found in the existing literature. So a new multi-objective performance metric (average fuzzy membership function) is proposed and used

https://doi.org/10.1017/S0263574708004359 Published online by Cambridge University Press

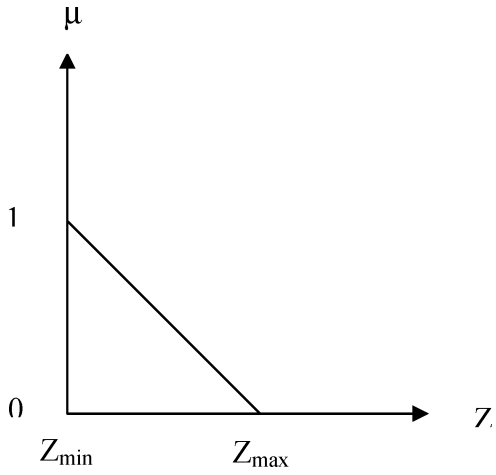

Fig. 7. Representation of fuzzy membership function for minimization of an objective function.

here to select a best optimal solution from the Pareto optimal fronts.

The membership function given in Fig. 7 is a graphical representation of the magnitude of each input. It associates a weighting value with each of the inputs that are processed, and determines an output response. These must be processed and combined in a manner to produce a single, crisp (defuzzified) output. The μ value is 1 at Z_{min} and 0 at Z_{max} for minimization of an objective function and it is vice versa for maximizing an objective function.

The proposed membership function of the fuzzy decision is as follows:

$\mu_i = (Z_{i\,max} – Z_i)/(Z_{i\,max} - Z_{i\,min})$ for minimization of objective function

$\mu_i = (Z_i – Z_{i\,min})/(Z_{i\,max} - Z_{i\,min})$ for maximization of objective function

Z_i–Objective function (i= Objective function number, 1. . . 6 for this problem)

Z_{max}–Maximum objective function value

Z_{min}–Minimum objective function value

The solution that has the highest average membership function value (μ_{avg}) is the best optimal solution, which gives a nondominated solution.

For our problem, the average membership function is defined as follows:

Maximize average membership function

$$(\mu_{avg}) = (\mu 1 + \mu 2 + \mu 3 + \mu 4 + \mu 5 + \mu 6)/6.0 \qquad (29)$$

where, μ1 = (z1max-z1)/(z1max-z1min), μ2 = (z2max-z2)/(z2max-z2min),

μ3 = (z3max-z3)/(z3max-z3min), μ4 = (z4max-z4)/(z4maxz4min),

μ5 = (z5max-z5)/(z5max-z5min), μ6 = (z6-z6min)/reak (z6max-z6min).

4.2. Solution spread measure

While it is desirable to find more Pareto-optimal solutions, it is also desirable to find the ones scattered uniformly over the Pareto frontier in order to provide a variety of compromise solutions to the decision maker. Solution Spread Measure (SSM) represents the distribution of the solutions along the Pareto front.

$$\text{SSM} = \frac{d_f + d_l + \sum_{i=1}^{N-1} |d_i - \bar{d}|}{d_f + d_l + (N-1)\bar{d}} \qquad (30)$$

where N is the number of solutions along the Pareto front so there are (N-1) consecutive distances, d_i is the distance (in objective space) between each solution, $\bar{d}$ is the arithmetic mean of all $d_i$and d_f and d_l are the Euclidean distances between the extreme solutions and the boundary solutions of the obtained non-dominated set. Thus, a low performance measure characterizes an algorithm with good distribution capacity.

4.3. Ratio of non-dominated individuals

This performance metric is defined as the ratio of non-dominated individuals (*RNI*) for a given population X,

$$RNI(X) = nondom\_indiv/P \qquad (31)$$

where *nondom_indiv* is the number of non-dominated individuals in population X and P is the size of population X. therefore the value $RNI = 1$ means all the individuals in the population are non-dominated, and $RNI = 0$ represents the situation where none of the individuals in the population are non-dominated. Since a population size of more than zero is often desired, there is always at least one non-dominated individual in the population within the range of $0 < RNI < 1$.

4.4. Optimiser overhead

Total number of evaluations and total CPU time may be used for testing the algorithm. This would be useful in indicating how long an optimisation or simulated evolution process would take in real world and to indicate the amount of program overhead as a result of the optimisation manipulations such as those by Evolutionary Algorithm operators. More quantitatively, the Optimiser Overhead (OO) may be calculated by

$$\text{Optimiser Overhead} = (T_{Total} - T_{PFP})/T_{PFP} \qquad (32)$$

Where T_{Total} is the total time taken and T_{PFP} is the time taken for pure function evaluations. Thus, a value of zero indicates that an algorithm is efficient and does not have any overhead. However, this is an ideal case and is not practically reachable.

4.5. Algorithm effort

The performance in multi-objective optimisation is often evaluated not only in terms of how the final pareto-front is, but also in terms of the computational effort required in obtaining the optimal solutions. For this purpose, the algorithm effort is defined as the ratio of the total number of functions evolutions N_{eval} over a fixed period of simulation time T_{run},

$$\text{Algorithm effort} = T_{run}/N_{eval},\ (T_{run} > T_{1stgen}) \cap (T_{eval} \propto N_{eval}) \qquad (33)$$

https://doi.org/10.1017/S0263574708004359 Published online by Cambridge University Press

Table I. Denavit–Hartenberg parameters for a PUMA 560 robot[5].

Joint No.	1	2	3	4	5	6
θ_i (°)	q_1	q_2	q_3	q_4	q_5	q_6
α_i (°)	–90	0	–90	90	–90	0
a_i (m)	0	0.4318	0.0203	0	0	0
d_i (m)	0	0	0.1254	0.4318	0	0

As shown in the above equation, for a fixed period of T_{run}, more number of function evolutions being performed indirectly indicates that less computational effort is required by the optimisation algorithm and hence resulting in a smaller algorithm effort. The condition of $T_{run} > T_{1stgen,}$ where T_{1stgen} is the computation time for the 1st generation, should be hold that T_{run} and N_{eval} are > 0. This results algorithm effort is bounded in the range of (0, ∞).

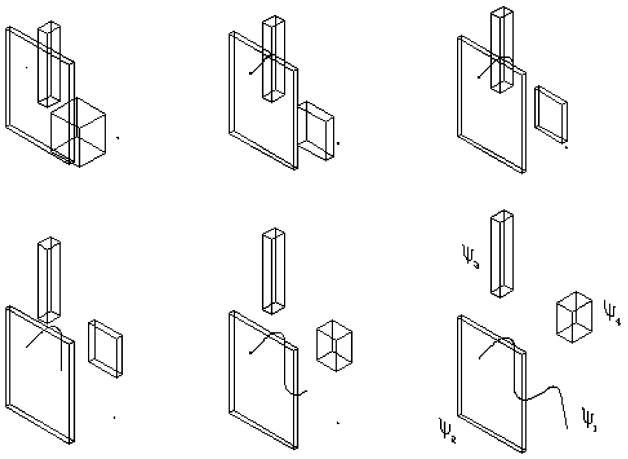

Fig. 8. End-effector tridimensional optimal trajectory.

5. Numerical Example

In this paper the proposed Multi-objective Differential evolution (MODE) and Elitist Non-dominated Sorting Genetic Algorithm (NSGA-II) are used for optimal trajectory planning of the PUMA 560 robot manipulator in material handling operation (Fig. 1). The Tables I and II represent the geometric and inertial parameters of PUMA 560 robot.[5] It is considered that $\dot{q}_1 = \dot{q}_m = \ddot{q}_1 = \ddot{q}_m = 0$ for all joints. Table III represents the constraints for joints displacement, velocity, acceleration, jerk, and force/torque.[5]

Five-knot points and four control points in each segment have been considered here. So there are 288 variables in this problem. In this application the aim is to obtain optimal trajectory (ψ_1) of the end-effectors as in Fig. 8 considering the following obstacles: A wall (ψ_2), a translating body (ψ_3), and a rotating and translating body (ψ_4).

Obstacle dimensions are: prism 1 ($X_1 = 0.5$ and $X_2 = 1.0$; $Y_1 = 1.0$ and $Y_2 = 1.4$; $Z_1 = 0.0$, $Z_2 = 1.0$), prism 2 ($X_1 = -0.8$ and $X_2 = 0.0$; $Y_1 = -2.0$ and $Y_2 = 1.0$; $Z_1 = 0.0$, $Z_2 = 0.5$) and prism 3 (wall) ($X_1 = -1.0$ and $X_2 = -0.9$; $Y_1 = -2.0$ and $Y_2 = 2.0$; $Z_1 = 0.0$, $Z_2 = 1.0$). The initial and final trajectory points of the end-effectors are: $q_1 = [0.1745\text{rd}, 0.1745\text{rd}, 0.80\,\text{m}, 0.0873\text{rd}, 0.1745\text{rd}, 0.1047\text{rd}]$ and $q_m = [-1.745\text{rd}, 1.396\text{rd}, 1.20\,\text{m}, -0.853\text{rd}, 1.078\text{rd}, -1.3894\text{rd}]$.

The wall is circumscribed by an ellipsoid and spheres circumscribe the moving bodies. Fig. 8 shows the end-effectors tridimensional trajectory.

The grasped object mass (payload) is 1 kg. The payload constraints are expressed in terms of feasible range for the grasping forces given by $Fg_{\min} = 0$ and $Fg_{\max} = 60\text{N}$.

The coefficients associated with dissipation force [Eq. (12)] are adopted as

f_c (Nm)=[0.058,0.058,0.058,0.056,0.056,0.056] and

f_d (Nm/s)=[0.0005,0.0005,0.000472,0.000382,0.000382, 0.000382].

Table II. Geometric and inertial parameters of PUMA560 robot[5].

Joint No.	1	2	3	4	5	6
M (kg)	–	18.5	4.8	0.94	0.54	0.10
r_x (m)	–	0.068	0	0	0	0
r_y (m)	–	0.006	−0.070	0	0	0
r_z (m)	–	−0.016	0.014	−0.019	0	0.032
I_{xx} (kg m^2)	0	0.13	0.066	1.8e-3	0.3e-3	0.15e-3
I_{yy} (kg m^2)	0	0.524	0.0125	1.8e-3	0.3e-3	0.15e-3
I_{zz} (kg m^2)	0.35	0.539	0.086	1.3e-3	0.4e-3	0.04e-3
I_m (kg m^2)	1.140	4.710	0.830	0.200	0.179	0.193

Table III. Limiting parameters used for Puma 560 robot.[5].

Joint No.	1	2	3	4	5	6
q^{max} (degree)	320	250	270	280	200	532
$\dot{q}^{\max}$ (degree/s)	82	74	122	228	241	228
$\ddot{q}^{\max}$ (degree/s^2)	286	286	572	572	572	572
$q^{\cdots max}$ (degree/s^3)	60	60	60	60	60	60
τ^{max} (Nm)	77	133	66	13	12	13

https://doi.org/10.1017/S0263574708004359 Published online by Cambridge University Press

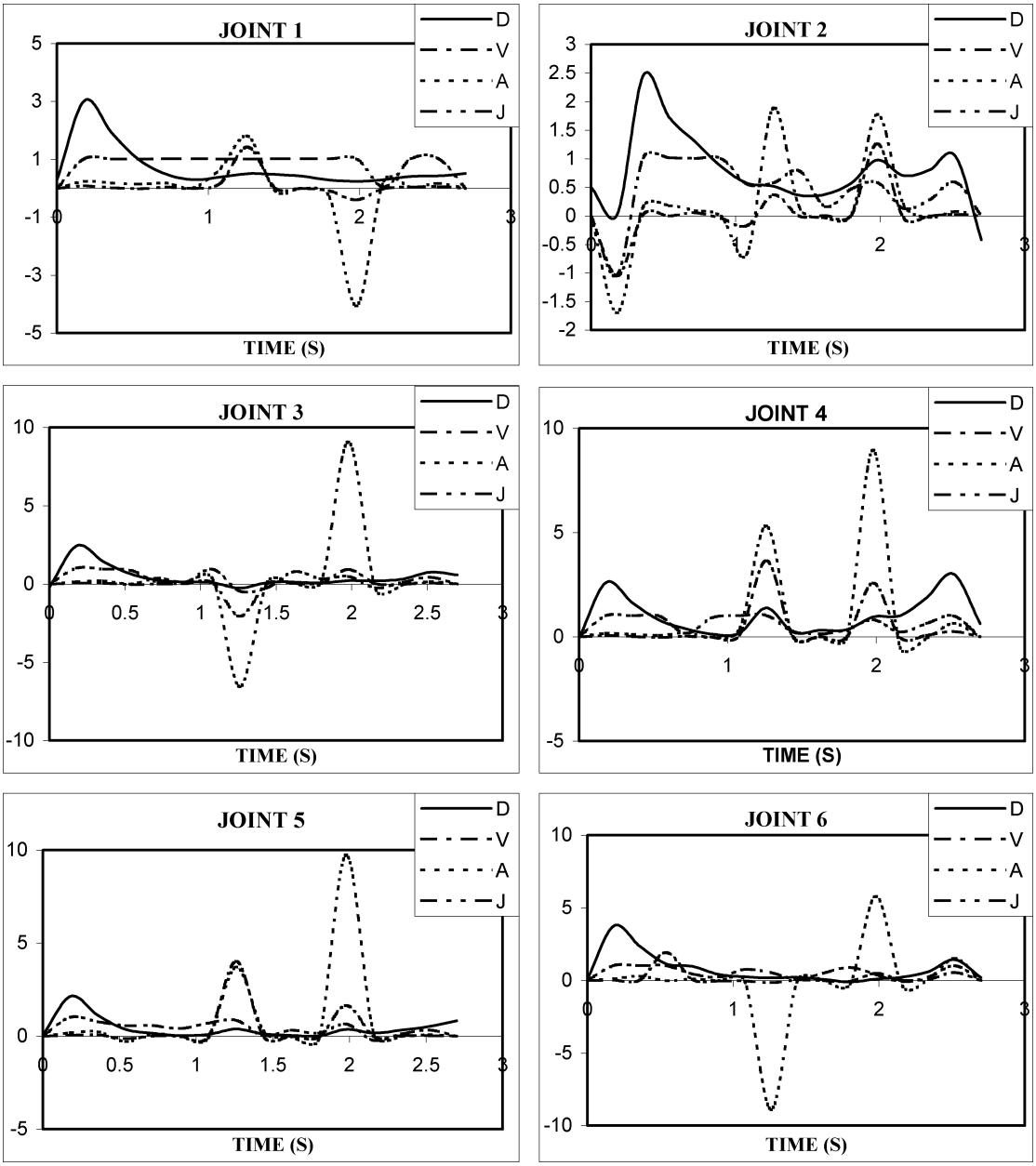

Fig. 9(a–d). Optimal motion obtained using MODE.

6. Results and Discussion

Figs 9 and 10 show the optimal displacement (D – rad), velocity (V – rad/s), acceleration (W – rad/s^2), and jerk (J – rad/s^3) of all the robot joints obtained from MODE and NSGA-II respectively. From Figs. 9 and 10, it is noted that the robot joints displacement, velocity, acceleration, and jerk are within their limiting values. The optimal solution trade-offs (Pareto optimal fronts) obtained from NSGA-II and MODE are given in Figs. 11 and 12 respectively. From Figs. 11 and 12, it is noted that NSGA-II gives more number of optimal solution trade-offs than MODE. So NSGA-II is best for this problem, if the user wants more number of solutions for his choice. T1he best solution tradeoffs selected by average fuzzy membership function method from the optimal solution trade-offs obtained from the NSGA-II and MODE are shown in Fig. 13. From Fig. 13, it is noted that the MODE gives best results for five objective functions (Minimum values for z1, z2, z3, z4 and z5). Also the computational time to find optimumsolutions in MODEis 1/3rd of that of the NSGA-II. MODE is faster than the NSGA-II. So the MODE is best to the NSGA-II for this problem. But NSGA-II gives best result for z6. To maintain a similar condition for both NSGA-II and MODE while finding their computational effort, the simulation time is assumed as 2 sec for both algorithms.

The results of average fuzzy membership function value (μ_{avg}), solution spread measure (SSM), ratio of non-dominated individuals (RNI), optimiser overhead (OO), and algorithm effort obtained from NSGA-II and MODE are listed in Tables IV–VI.

The algorithm that gives maximum average membership function value (μ_{avg}), minimum solution spread measure (SSM), maximum ratio of non-dominated individuals (RNI),

https://doi.org/10.1017/S0263574708004359 Published online by Cambridge University Press

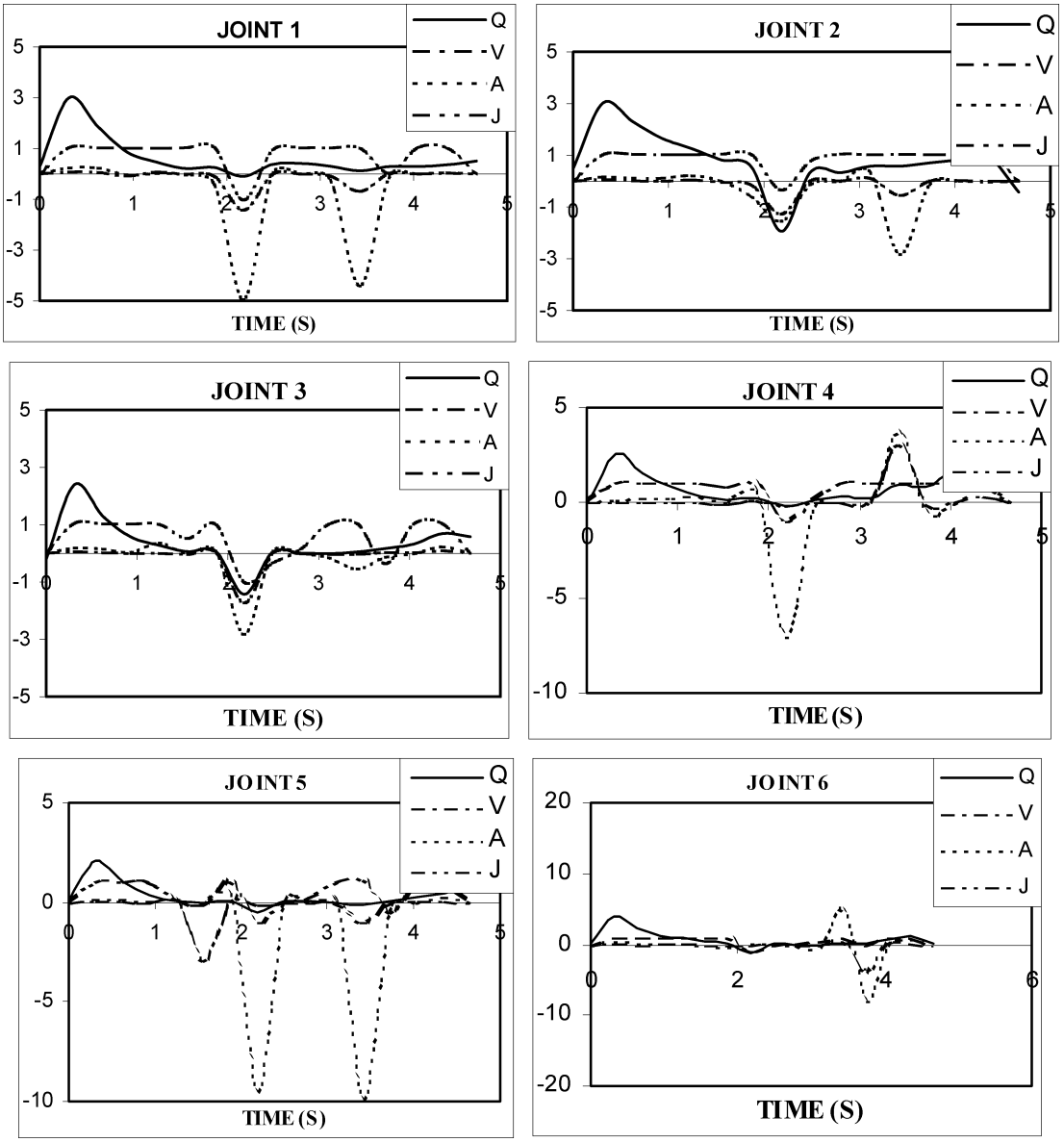

Fig. 10(a–d). Optimal motion obtained using NSGA-II.

minimum optimiser overhead (OO), and minimum algorithm effort is the best optimisation algorithm.

From Tables IV–VI, it is observed that the NSGA-II technique gives maximum average fuzzy membership function (μ_{avg}), minimum solution spread measure (SSM), and maximum ratio of non-dominated individuals (RNI) than those of MODE. But the MODE gives minimum optimiser overhead (OO) and minimum algorithm effort than those of the NSGA-II. So both NSGA-II and MODE are best for this problem.

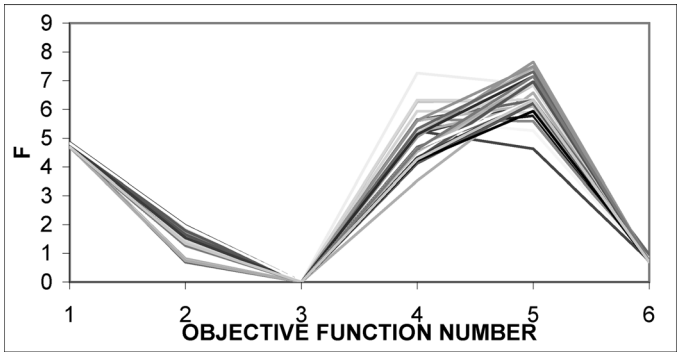

Fig. 11. Optimal solution tradeoffs obtained from NSGA-II.

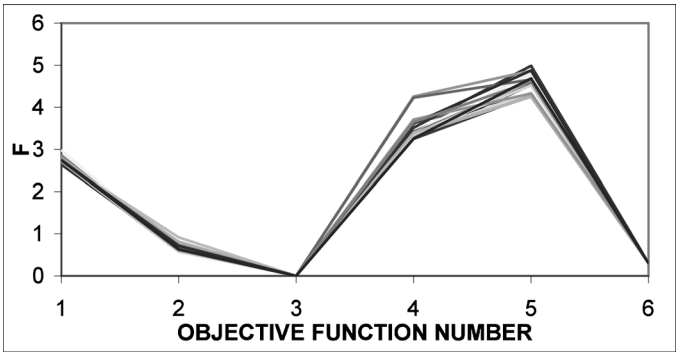

Fig. 12. Optimal solution tradeoffs obtained from MODE.

https://doi.org/10.1017/S0263574708004359 Published online by Cambridge University Press

Table IV. Results obtained from NSGA-II and MODE algorithms.

	z1	z2	z3	z4	z5	z6	μ_{avg}
Zmax	5.33	16754.5	0.0	578.614	564.328	0.725415	
Zmin	2.7	5798.847	0.0	325.47	451.114	0.300452	
NSGA-II	4.67	16062.375	0.0	565.410	558.135	0.703259	0.228142
MODE	2.7	5798.847	0.0	327.640	452.767	0.300452	0.662805

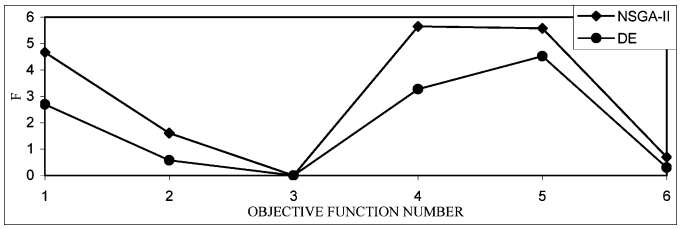

Fig. 13. Best solution tradeoffs obtained from the NSGA-II and MODE.

Table V. Algorithm effort obtained from NSGA-II and MODE algorithms.

Proposed Algorithm	Simulation time T_{run} (sec)	No. of function evolution (N_{eval})	Algorithm effort
NSGA-II	2	91	0.022
MODE	2	122	0.017

Table VI. Results of multiobjective performance metrics.

Proposed Algorithm	Solution spread measure (SSM)	Ratio of non-dominated individuals (RNI)	Optimiser overhead (OO)
NSGA-II	0.00105	0.35	0.1616
MODE	0.00132	0.27	0.0987

7. Conclusions

A general new methodology using NSGA-II and MODE for the off-line tridimensional optimal trajectory planning of the industrial robot manipulator (PUMA 560 robot) in the presence of fixed and moving obstacles with payload constraint is presented. Obstacle avoidance is obtained by adding penalty functions to the multicriterion problem. When dealing with fixed and moving obstacles all the objective functions and the constraint functions have to be updated simultaneously at each time instant. A cubic NURBS curve is used to define the trajectory. The average fuzzy membership function method is used to select best optimal solution from Pareto optimal fronts. Two multi-objective performance measures viz., solution spread measure and ratio of non-dominated individuals are used to evaluate strength of the Pareto optimal fronts. Two more multi-objective performance measures namely optimiser overhead and algorithm effort are used to find computational effort of NSGA-II and MODE algorithms. The Pareto optimal fronts (optimal solution trade-offs) and results obtained from various techniques are compared and analysed. The results indicate that MODE technique gives minimum optimiser overhead (OO) and minimum algorithm effort than those of NSGA-II i.e., it is faster than NSGA-II technique. Also the computational time to find optimum solutions in MODE is $1/3^{rd}$ of that of the NSGA-II. MODE is faster than the NSGA-II. So MODE is best to the NSGA-II for this problem, if the user wants a best optimal solution quickly. But NSGA-II technique gives maximum average fuzzy membership function (μ_{avg}), minimum solution spread measure (SSM), maximum ratio of non-dominated individuals (RNI) than those of MODE. Also NSGA-II gives the best Pareto optimal front with more number of non-dominated solutions for user's choice than MODE. So NSGA-II technique is best for this multicriterion optimisation problem. This work opens the door for further investigations on how the evolutionary optimisation techniques can be used to solve complex problems.

References

1. S. F. P. Saramago and V. Steffen Jr, "Optimization of the trajectory planning of robot manipulators taking into account the dynamics of the system," *Mech. Mach. Theory* **33**(7), 883–894 (1998).
2. S. F. P. Saramago and V. Steffen Jr, "Dynamic optimization for the trajectory planning of robot manipulators in the presence of obstacles," *J. Brazilian Soc. Mech. Sci.* **21**(3), 1–17 (1999).
3. S. F. P. Saramago and V. Steffen Jr, "Optimal trajectory planning of robot manipulators in the presence of moving obstacles," *Mech. Mach. Theory* **35**(8), 1079–1094 (2000).
4. S. F. P. Saramago and V. Steffen Jr, "Trajectory modeling of robot manipulators in the presence of obstacles," *J. Optim. Theory Appl.* **110**(1), 17–34 (2001).
5. S. F. P. Saramago and M. Ceccareli, "An optimum robot path planning with payload constraints," *Robotica* **20**, 395–404 (2002).
6. T. Chettibi, H. E. Lehtihet, M. Haddad and S. Hanchi, "Minimum cost trajectory planning for industrial robots," *European J. Mech. A/Solids* **23**, 703–715 (2004).
7. Aurelio Piazzi, "Global minimum-jerk trajectory planning of robot manipulators," *IEEE Trans. Ind. Electron.* **47**(1), 140–149 (2000).
8. A. Gasparetto and V. Zanotto, "A new method for smooth trajectory planning of robot manipulators," *Mech. Mach. Theory* **42**(4), 455–471 (2007).
9. A. Elnagar and A. Hussein, "On optimal constrained trajectory planning in 3D environments," *Robot. Auton. Syst.* **33**(4) 195–206 (2000).
10. J. E. Lloyd and Vincent Hayward, "Singularity-robust trajectory generation," *Int. J. Robot. Res.* **20**(1), 38–56 (2001).
11. Chih-Jer Lin, "Motion planning of redundant robots by perturbation method," *Mechatronics* **14**, 281–297 (2004).
12. N. Tatematsu and K. Ohnishi, "Tracking Motion of Mobile Robot for Moving Target Using NURBS Curve," *Proceedings of IEEE Int. Conference on Ind. Technol.*, **1** (2003), Maribor, Slovenia, pp. 245–249.
13. A. A. Ata and R. T. Myo, "Optimal point-to-point trajectory tracking of redundant manipulators using generalized pattern search," *Int. J. Adv. Robot. Syst.* **2**, 239–244 (2005).

https://doi.org/10.1017/S0263574708004359 Published online by Cambridge University Press

14. K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.* **6**(2), 182–197 (2002).
15. B. V. Babu and B. Anbarasu, "Multi-Objective Differential Evolution (MODE): An Evolutionary Algorithm for Multi-Objective Optimization Problems (MOOPs)," http://discovery.bitspilani.ac.in/discipline/chemical/BVb/publications/htmlPrice.
16. K. Price and R. Storn, "Differential Evolution—A simple evolution strategy for fast optimisation," *Dr. Dobb's J.* **22**(4), 18–24, 78 (1997).
17. L. Piegl and W. Tiller, "The NURBS Book," 2nd edn. (Springer, Berlin Heidelberg New York, 1997).

Appendix

The Pseudo code for MODE algorithm is presented below:

The following assumes that we are minimizing all the objective functions, f_q

(1) Generate box, P, of N_p parent vectors using a random-number code to generate the several real variables. These vectors are given a sequence (position) number as generated
(2) Classify these vectors into fronts based on nondomination as follows:
  (a) Create new (empty) box, P', of size, N_p
  (b) Transfer i^{th} vector from P to P', starting with I = 1
  (c) Compare vector I with each member, say each member, say j, already present in P', one at a time
  (d) If i dominates over j (i.e., i is superior to or better than j in terms of all objective functions), remove the j^{th} vector from P' and put it back in its original location in P
  (e) If i dominated over by j, remove i from P' and put it back in its position in P
  (f) If i and j are non-dominating (i.e. there is at least one objective function associated with i that is superior to/better than that of j), keep both i and j in P' (in sequence). Test for all j present in P'
  (g) Repeat for next vector (in the sequence, without going back) in P till all N_p are tested. P' now contains a sub-box (of size $< = N_p$) of nondominated vectors (a subset of P), referred to as the first front or sub-box. Assign it a rank number, I_{rank}, of I
  (h) Create subsequent fronts in (lower) sub-boxes of P', using Step 2b above (with the vectors remaining in P). Compare these members only with the members present in the current sub-box, and not with those in earlier (better) sub-boxes. Assign these I_{rank} = 2,3. . . Finally, we have all N_p vectors in P', boxed into one or more fronts.
(3) Spreading out: Evaluate the crowding distance, $I_{i,dist}$, for the i^{th} vector in any front, j, of P' using the following procedure:
  (a) Rearrange all vectors in front j in ascending order of the values of any one (say, the qth) of their several objective functions (fitness functions). This provides a sequence, and, thus, defines the nearest neighbors of any vector in front j.
  (b) Find the largest cuboid (rectangle for two fitness functions) enclosing vector i that just touches its nearest neighbors in the f-space.
  (c) $I_{i,dist}$ = $^1/_2$*(sum of all sides of this cuboid)
  (d) Assign large values of Ii, dist to solutions at the boundaries (the convergence characteristics would be influenced by this choice).
(4) Perform DE operation over the NP target vectors in P' to generate NP trial vectors and store it in P".
  (a) (Create new (empty) box, P", of size, N_p
  (b) Select a target vector, i in P', starting with I = 1
  (c) Choose two vectors, r1 and r2 at random from the NP vectors in P' and find the weighted difference. This is carried out by the following steps: (1) Generate two random numbers, (2) decide which two population members are to be selected, (3) Find the vector difference between the two vectors. Multiply this difference with F to obtain the weighted difference.
  (d) Find the noisy random vector. This is done by (1) Generate a random number, (2) choose a third random vector, r3, from the NP vectors in P', (3) Add this vector to the weighted difference to obtain the noisy random vector.
  (e) Perform Crossover between the target vector and noisy random vector to find the trial vector and put it in box P". This is carried out by (1) Generate random numbers equal to the dimension of the problem, (2) For each of the dimensions: if random no. > CR; copy the value from the target vector, else copy the value from the noisy random vector into the trial vector and put it in box P".
(5) Elitism: Copy all the N_p parent vectors (P') and all the N_p trial vectors (P") into box PT. Box PT has $2N_p$ vectors
  (a) Reclassify these 2Np vectors into fronts (box PT') using only non-domination (as described in Step 2 above).
  (b.) Take the best N_p from box PT' and put into box P"'. The following procedure is adopted to identify the better of the two chromosomes. Chromosome i is better than chromosome j if
  $I_{i,rank} \neq I_{j,rank} : I_{i,rank} < I_{j,rank}$
  $I_{i,rank} = I_{j,rank} : I_{i,dist} > I_{j,dist}$
  This completes one generation. Stop if appropriate criteria are met, e.g. the generation number > maximum number of generations (user specified). Else, Copy P"' into starting box P. Go to Step 2 above.

https://doi.org/10.1017/S0263574708004359 Published online by Cambridge University Press