Robotica (2024), **42**, pp. 2026–2045
doi:10.1017/S0263574724000766

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

Multi-objective trajectory planning for industrial robots using a hybrid optimization approach

Taha Chettibi

Structures Laboratory, Department of Mechanics, Saad Dahlab University, Blida, Algeria
Email: chettibi_taha@univ-blida.dz

Received: 1 March 2024; **Accepted:** 16 April 2024; **First published online:** 10 May 2024

Keywords: trajectory planning; robot manipulator; spline; hybrid algorithm; multi-objective optimization

Abstract

In this paper, a hybrid approach organized in four phases is proposed to solve the multi-objective trajectory planning problem for industrial robots. In the first phase, a transcription of the original problem into a standard multi-objective parametric optimization problem is achieved by adopting an adequate parametrization scheme for the continuous robot configuration variables. Then, in the second phase, a global search is performed using a population-based search metaheuristic in order to build a first approximation of the Pareto front (PF). In the third phase, a local search is applied in the neighborhood of each solution of the PF approximation using a deterministic algorithm in order to generate new solutions. Finally, in the fourth phase, results of the global and local searches are gathered and postprocessed using a multi-objective direct search method to enhance the quality of compromise solutions and to converge toward the true optimal PF. By combining different optimization techniques, we intend not only to improve the overall search mechanism of the optimization strategy but also the resulting hybrid algorithm should keep the robustness of the population-based algorithm while enjoying the theoretical properties of convergence of the deterministic component. Also, the proposed approach is modular and flexible, and it can be implemented in different ways according to the applied techniques in the different phases. In this paper, we illustrate the efficiency of the hybrid framework by considering different techniques available in various numerical optimization libraries which are combined judiciously and tested on various case studies.

1. Introduction

Robotic manipulators play a crucial role in various industrial and service applications, by helping to improve efficiency, productivity, quality control, and innovation. A key step in the exploitation process of these machines is trajectory planning [1]. It consists in determining the time history of various robot kinodynamic parameters for the control unit, ensuring the appropriate execution of the desired task. This process is generally performed by solving an appropriate optimization problem in order to improve specific robot performances.

Early papers dealing with the robot's trajectory planning problem (TPP) were oriented toward enhancing productivity by minimizing exclusively the execution time [2–4]. They were rooted in the field of optimal control theory, which provided powerful tools to generate optimal trajectories when high-speed motion is desired. Numerous numerical methods have been proposed to solve this class of problems. They can be grouped into two families: direct and indirect methods [5, 6]. In general, indirect methods are numerical solutions of Pontryagin's maximum principal optimality conditions. However, such techniques suffer from several drawbacks mainly, and they need to solve a difficult nonlinear multi-point shooting problem involving the use of co-state variables that are, in general, quite difficult to estimate [7–10]. Direct methods have been suggested to overcome some of these disadvantages, and they make use of different discretization schemes for state and control variables to transform the original infinite-dimensional TPP to a more tractable finite-dimensional nonlinear parametric optimization

problem that can be solved using classical mono-objective optimization techniques [5, 6]. Both nonlinear deterministic programming techniques [9–12] and stochastic optimization techniques [13–17] were successfully applied to compute suboptimal trajectories for the considered robots.

The above-mentioned studies focused mainly on solving TPP for a single objective. However, in real-world applications, multiple conflicting objectives, such as minimizing cycle time, maximizing energy efficiency, reducing vibration, or minimizing joint torques, might be involved in the trajectory planning process [18]. Thus, the TPP should be handled as a multi-objective optimization problem (MOOP) where a set of compromise solutions optimizing simultaneously the considered cost functions are investigated. These solutions constitute in the objective space what we call a Pareto front (PF) [19, 20], from which the decision-maker can select the relative best robot trajectory.

Many techniques and approaches have been proposed over the past four decades to solve MOOPs, and they can be classified according to when the preferences of the decision-maker are inserted in the solution process; thus, we have a priori, interactive, a posteriori method, or no preference methods [19, 21]. Also, they can be categorized, according to the method used for handling the vector of objective functions, into scalarization methods and vectorization methods. Scalarization methods convert the MOOP into a series of parametric single-objective optimization problems which are then solved using standard mono-objective optimization techniques [22]. For example, in refs. [12, 23], authors used the weighted sum method to solve TPP involving different cost functions. Unfortunately, scalarization methods are generally regarded as local optimization approaches and suffer from a loss of information about the relative importance of the different objectives.

Vectorization methods are handling directly the original MOOP and work directly with the PF composed of all nondominated solutions. Pareto-based methods are population-based derivative-free methods, so no-gradient information is needed, and they use in general nature-inspired metaheuristics. They belong to different classes such as swarm intelligence, evolutionary computation, physics- and chemistry-based algorithms, or evolutionary strategy [24, 25]. They are generally simple to implement and regarded as global optimization approaches. This explains the recent interest of researchers in solving the TPP for robotic manipulators using this class of algorithms. For example, Saravanan *et al.* [26, 27] studied two evolutionary algorithms, multi-objective differential evolution algorithm (MODE) and nondominated sorting genetic algorithm (NSGA-II) to treat the TTP for serial robots. In ref. [28], a constrained multi-objective particle swarm algorithm was applied to solve the TTP in order to minimize the traveling time, the energy, and the distance of the end effector. Also, Shi *et al.* [29] proposed a method based on NSGA-II to optimize the robot trajectories for three objectives, namely time, energy, and jerk. The performances of MODE and NSGA-II in processing the trajectory optimization problem were also investigated in ref. [30]. In ref. [31], the TPP is solved by using NSGA-II for two objectives, traveling time and mean jerk along the whole trajectory.

More recently, an improved multi-objective ant-lion algorithm is presented in ref. [32] to handle the time–jerk–torque optimal problem for a six-axis robot. In ref. [33], the trajectory competitive multi-objective particle swarm optimization algorithm is used to search for the optimal PF for a collaborative robot. Three performance criteria were considered: total motion time, average acceleration, and average jerk. In ref. [34], authors treated also the TPP for serial manipulators passing through waypoints by minimizing execution time, energy consumption, and joint jerks using NSGA-II. In ref. [35], authors presented a method to solve the minimum time–jerk TPP for serial manipulators in the presence of obstacles using a competitive mechanism-based multi-objective particle swarm optimizer and the Z-type fuzzy membership function is proposed to select the best solutions. Cheng *et al.* [36] utilized NSGA-II to solve the problems of inefficient transcranial magnetic stimulation manipulator execution and uncontrollable head safety collision, by optimizing the arm trajectories, which effectively improved the operation efficiency and protected the human head from injury. In ref. [37], authors proposed a many-objective trajectory optimization method based on response surface methodology and NSGA-III to optimize the motion of a masonry robot taking into account numerous objectives and constraints.

The analysis of the previous papers indicates that most multi-objective robot TPPs were solved using population-based nature-inspired metaheuristics and proposed approximations of the PF regrouping

https://doi.org/10.1017/S0263574724000766 Published online by Cambridge University Press

compromise solutions. However, these algorithms do not guarantee convergence to the true PF because of the inherent stochastic- and heuristic-based search procedures. In fact, these algorithms, in general, mimick the natural process of evolution and operate by using stochastic operators such as crossover, breeding, mutation, and selection to improve the nondominated solutions [38, 39]. Thus, most of these algorithms require high computation time to converge and have a slow convergence speed to the true PF. Moreover, there is a lack of theoretical convergence proof to the true optimal PF, and it is hard to determine the appropriate stopping criterion [40, 41]. Researchers often define the iterations as the termination condition. In consequence, in order to get a satisfactory result, the number of iterations is always defined as large, which means that computation time and efficiency are unacceptable. In order to overcome these drawbacks and make the overall search procedure faster in a more guaranteed manner, it is strongly recommended to use hybrid algorithms combining population-based metaheuristics with mathematical optimization techniques having better convergence properties [20, 42–44].

In this paper, we propose to use a hybrid approach to solve the multi-objective optimization TPP. It combines population-based nature-inspired algorithms with deterministic algorithms to find a better approximation of the true optimal PF. It is organized in four successive phases. The first phase is a transcription step where the original problem is transformed into a standard MOOP by adopting an adequate parametrization scheme for the continuous robot configuration variables. In the second phase, at least one population-based metaheuristic is used to perform a global search in the solution space and reach the region near the optimal PF in a relatively small number of generations. In the third phase, a deterministic algorithm is used to enhance the quality of found solutions by performing a local search starting from each solution of the approximated PF according to an adequate scalarization method. Finally, in the fourth phase, results of the global and local searches are gathered and postprocessed using a multi-objective direct search method to enhance the quality of compromise solutions and to converge toward the true optimal PF. This judicious combination of vectorization and scalarization methods balances their exploration and exploitation capabilities to converge toward a diverse and high-quality PF. Indeed, by using various optimization techniques having different mechanisms to explore and exploit the search space, the resulting hybrid algorithm ensures not only improvement of the overall search mechanism but also the resulting high-level relay hybrid algorithm should keep the robustness of the nature-inspired algorithm while enjoying the theoretical properties of convergence of the deterministic component. It is worth noting that the proposed approach can be implemented in different ways according to the techniques being used at the successive phases. In order to illustrate the flexibility of the proposed approach, we propose to use, in the first phase, spline functions of different orders to interpolate a set of control nodes in order to generate trajectory candidates. The coordinates of these nodes become the optimization variables of the transformed problem. In the following phases, different optimization algorithms can be used, and they are available in various numerical optimization libraries such as the Matlab Global Optimization Toolbox [45], the object-oriented framework for engineering optimization EngiO [46], the open-source platform for solving optimization problems PlatEMO [38, 47], or pymoo [39]. The efficiency of the proposed hybrid approach is illustrated by solving a set of test cases. The remainder of this paper is organized as follows. In Section 2, the problem formulation of the original TPP is detailed. In Section 3, the proposed approach is developed with its four phases. Section 4 regroups different numerical examples treating the cases of two-degrees-of-freedom (*dof*) and six-*dof* robots. Finally, Section 5 concludes this work.

2. Problem formulation

Let us consider a serial manipulator with *n dof* required to move from an initial location defined by $\boldsymbol{q}^{ini}$ to a final location $\boldsymbol{q}^{fin}$. The equation of motion for the i^{th} joint can be written as follows [1]:

$$\sum_{j=1}^{n} M_{ij}\left(\boldsymbol{q}\left(t\right)\right)\ddot{q}_j\left(t\right) + C_i\left(\boldsymbol{q}\left(t\right), \dot{\boldsymbol{q}}\left(t\right)\right) + G_i\left(\boldsymbol{q}\left(t\right)\right) = \Gamma_i\left(t\right) \tag{1}$$

https://doi.org/10.1017/S0263574724000766 Published online by Cambridge University Press

where $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$ are, respectively, vectors of joint velocity and acceleration. $\boldsymbol{M}(\boldsymbol{q})$ is the inertia matrix and $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is the vector of centrifugal and Coriolis forces. $G(\boldsymbol{q})$is the vector of potential forces and $\Gamma(t)$ is the vector of input actuators efforts.

The multi-objective TPP for a robotic manipulator aims to program the robot to move from an initial configuration $\boldsymbol{q}^{ini}$ to a final configuration $\boldsymbol{q}^{fin}$, with optimum performances, while a set of kinematic and dynamic constraints are respected. Thus, the problem consists of determining the transfer time T, the joint trajectory vector $\boldsymbol{q}(t)$ and its time derivatives as well as the corresponding actuator efforts $\Gamma(t)$ such that all constraints are respected and a vector of cost functions $\boldsymbol{F}_{\mathbf{obj}}$ is optimized.

2.1. Constraints

2.1.1 Boundary conditions

Without loss of generality, we suppose that the limit configurations ($\boldsymbol{q}^{ini}$, $\boldsymbol{q}^{fin}$) are characterized by null joint velocities and accelerations. Thus, we have

$$\boldsymbol{q}(t=0) = \boldsymbol{q}^{ini}; \qquad \boldsymbol{q}(t=T) = \boldsymbol{q}^{fin} \tag{2a}$$

$$\dot{\boldsymbol{q}}(t=0) = 0; \qquad \dot{\boldsymbol{q}}\,(t=T) = 0 \tag{2b}$$

$$\ddot{\boldsymbol{q}}\,(t=0) = 0; \qquad \ddot{\boldsymbol{q}}\,(t=T) = 0; \tag{2c}$$

2.1.2 Geometric constraints

First, there are constraints imposed on joint positions:

$$q_i^{min} \le q_i(t) \le q_i^{max} \quad i = 1, \ldots, n \quad 0 \le t \le T \tag{3a}$$

But, if obstacles are present in the workspace, collisions must be avoided and the following additional constraint will hold during the transfer:

$$B(\boldsymbol{q}(t)) = False \qquad 0 \le t \le T \tag{3b}$$

Here, B denotes a Boolean function that indicates whether the robot at configuration $\boldsymbol{q}$ is in collision either with an obstacle or with itself.

2.1.3 Kinematic constraints

During the robot motion and depending on the problem at hand, limitations may be imposed on the following kinematic parameters: for $i = 1, \ldots, n$

- joint velocities: $|\dot{q}_i(t)| \le \dot{q}_i^{max}$ (4a)
- joint accelerations: $|\ddot{q}_i(t)| \le \ddot{q}_i^{max}$ (4b)
- joint jerks: $|\dddot{q}_i(t)| \le \dddot{q}_i^{max}$ (4c)

These constraints define bounds ($\dot{\boldsymbol{q}}^{max}, \ddot{\boldsymbol{q}}^{max}, \dddot{\boldsymbol{q}}^{max}$) on the robot kinematics performances due to technological restrictions or to the nature of the assigned task. Of course, nonsymmetrical bounds can be treated also.

2.1.4 Dynamic constraints

In regard to the robot's actuators' capacities, limits can be also imposed on the amplitude of the input joint torques, such as:

$$|\Gamma_i(t)| \le \Gamma_{\mathrm{i}}^{\max} \qquad i = 1, \ldots, n \tag{5}$$

https://doi.org/10.1017/S0263574724000766 Published online by Cambridge University Press

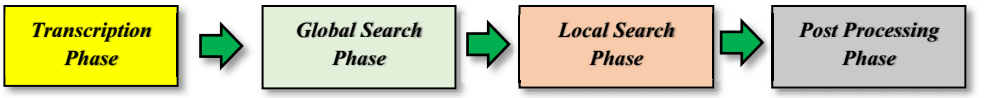


Figure 1. *A four-phase multi-objective trajectory planning approach.*

2.2. Objective functions

The vector $\boldsymbol{F}_{obj}$ of objective functions regroups the desired m cost functions to be optimized simultaneously during the task achievement. Each component of $\boldsymbol{F}_{obj}$ represents a significant physical quantity related to the robot's behavior or the productivity of the robotized task. From the perspective of practical applications of robotic manipulators, the i^{th} $(i = 1,\ldots,m)$ component of the vector $\boldsymbol{F}_{obj}$ might be represented by one of the following expressions:

- minimum time: $$J_T = \int_0^T dt = T \tag{6a}$$
- minimum Jerk: $$J_J = \int_0^T \sum_{i=1}^{n} \left(\frac{\dddot{q}_i(t)}{\dddot{q}_i^{max}} \right)^2 dt \tag{6b}$$
- minimum efforts: $$J_E = \int_0^T \sum_{i=1}^{n} \left(\frac{\Gamma_i(t)}{\Gamma_i^{max}} \right)^2 dt \tag{6c}$$
- minimum power: $$J_P = \int_0^T \sum_{i=1}^{n} \left(\frac{\dot{q}_i \Gamma_i(t)}{\dot{q}_i^{max} \Gamma_i^{max}} \right)^2 dt \tag{6d}$$

The TPP defined by relations (2–6) is a complex multiple objective optimal control problem. It needs to be first transformed into a standard MOOP by adopting an appropriate approximation for the joint trajectory vector $\boldsymbol{q}(t)$. The resulting MOOP can be then treated using a hybrid multi-objective optimization algorithm as explained in the following sections.

3. Proposed approach

The proposed approach includes four phases (Fig. 1). The first phase is a transcription step where the original TPP is transformed into a standard MOOP by adopting an adequate parametrization scheme for the robot configuration variables. Thus, instead of looking for a set of continuous functions representing the robot state or its input controls, the optimization process searches for a set of discrete parameters. After that and based on the transformed TPP formulation, the optimization process is performed in the three following phases that constitute together a hybrid multi-objective algorithm. A hybrid multi-objective algorithm aims to capitalize on the strengths of different algorithms, leading to an improved convergence, a better exploration of the solution space, and a more effective exploitation of the inherent problem characteristics. Thus, in the second phase of the proposed approach, a population-based search technique is used to globally explore the solution space and reach the region near the targeted optimal PF in a relatively small number of generations. Then, in the third phase, a deterministic optimization technique is used to search iteratively for new nondominated points by examining the neighborhood of solutions obtained in the second phase. Finally, solutions obtained in the second and third phases are gathered in a unique population and postprocessed in the fourth phase in order to remove possible non-Pareto optimal points and to enhance the quality of the final PF.

https://doi.org/10.1017/S0263574724000766 Published online by Cambridge University Press

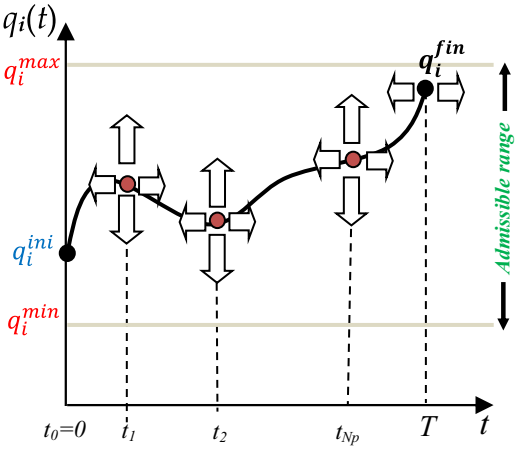


Figure 2. *Approximation of a joint variable $q_i(t)$ using N_p nodes.*

In fact, different schemes of hybridization, according to low-level schemes or high-level schemes, have been proposed in the literature and showed to be more effective than using pure metaheuristics [42, 43, 48, 49]. In a low-level scheme, a given function of a metaheuristic is replaced by another algorithm. In a high-level scheme, there is no composition of the applied metaheuristics, retaining their own identities and working as a relay, where one metaheuristic takes as its inputs the output of the precedent metaheuristic, working in series and using cooperative optimization models.

In this paper, we are going to use different optimization techniques which are available in the Matlab Global Optimization Toolbox [45] and PlatEMO [38, 47]. More precisely, a global–local search strategy based on a high-level relay hybrids model with a sequential execution order will be adopted [43]. We propose to use first, at least, one population-based algorithm for obtaining a diversified approximation of PF. Then, this approximation is improved by applying a local deterministic search on each solution of the PF approximation. Finally, non-Pareto optimal points are eliminated and the quality of the final PF is enhanced by applying a direct search multi-objective algorithm. In the proposed hybrid optimization framework, the population-based algorithm plays the role of a global optimizer by searching the entire search space to find the most promising regions with a population of individuals, while a local search module locally improves the individuals of a population. The final algorithm acts like a Pareto filter.

3.1. Problem transcription

The first phase of the proposed approach aims to convert the TPP defined by relations (2–6) which is a complex multiple objective optimal control problem, into a standard MOOP. This will be done by adopting an appropriate joint motion approximation. Thus, any trajectory candidate $\boldsymbol{q}(t)$ will be represented using n distinct sets of nodes: one set of N_p nodes for each joint variable (Fig. 2). The first and last nodes are fixed according to boundary conditions (2a), but the free interior nodes can be randomly positioned to produce a trial joint trajectory $q_i(t)$ using any appropriate fitting model that accounts for conditions (2b) and (2c). Indeed, each interior node can be moved horizontally along the timescale and vertically within the admissible range defined by relation (3a).

The fact that $q_i(t)$ must go exactly through the intermediate nodes means that we have at hand an interpolation problem. One straightforward way to generate smooth curves for a given interpolation point is to use splines. There are two commonly used methods to represent a spline function, the piecewise polynomial form (PP-form) and the basis B-spline form (*B*-form) [50, 51]. For example, a k^{th} order spline in PP-form, for the strict increasing break sequence $t_0 = 0, t_1, \ldots, t_{Np+1} = T$, is by definition of

https://doi.org/10.1017/S0263574724000766 Published online by Cambridge University Press

any function representing $q_i(t)$, that on each of the intervals $[t_i, t_{i+1}]$ agrees with some polynomial of degree less than k. The function $q_i(t)$ has in consequence continuous derivatives up to order (k-2). The main advantage of using spline functions is that it can provide high-degree accuracy while still being computationally efficient. Also, programming a robot task reduces to finding the best positions of the via nodes of n spline functions in order to build a trajectory candidate $\boldsymbol{q}(t)$, $0 \leq t \leq T$, respecting all imposed kinodynamic constraints and minimizing a set of objective functions.

In what follows, we are going to adopt the spline model for representing the robot's joint trajectories $\boldsymbol{q}(t)$. Thus, the original infinite-dimensional optimal TPP can be cast as a classical MOOP as follows [19, 20]:

$$\min_{X} \boldsymbol{F}(\boldsymbol{X}) \tag{7}$$

$$\text{subject to}: \quad \boldsymbol{h}(\boldsymbol{X}) = 0, \quad \boldsymbol{g}(\boldsymbol{X}) \leq 0, \quad \boldsymbol{x}_{min} \leq \boldsymbol{X} \leq \boldsymbol{x}_{max}$$

where

- $\boldsymbol{X} = \{x_1, x_2, \ldots, x_p\} \in \mathbb{R}^p$ is the vector of decision variables inherent to the adopted discretization scheme, and it brings together the coordinates of the spline nodes. Note that the abscissa of the last node is the value of the transfer time and if the nodes are uniformly distributed along the timescale, the size of the problem is reduced considerably. The vectors $\boldsymbol{x}_{min}$ and $\boldsymbol{x}_{max}$ define the limits of the search space, and they are defined according to (3a) and a superior born on transfer time T is defined in order to limit the search space.
- $\boldsymbol{F}(\boldsymbol{X})$ is the vector of m objective functions $[f_1(\boldsymbol{X}), f_2(\boldsymbol{X}), \ldots, f_m(\boldsymbol{X})]$ representing a transcription of the original m cost functions defined by relations (6a-d). The performance vector $\boldsymbol{F}(\boldsymbol{X})$ maps the parameter space of $\boldsymbol{X}$ into the objective functions space. Also, in the previous formulation, pure minimization problems are considered. However, without the loss of generality, an objective which is supposed to be maximized can be multiplied by -1 and be minimized.
- The feasible region of any solution candidate $\boldsymbol{X}$ is delimited by a vector of equality constraints $\boldsymbol{h}(\boldsymbol{X})$ and a vector of inequalities constraints $\boldsymbol{g}(\boldsymbol{X})$ corresponding to the transcription of the original problem path constraints defined by relations (3–5).

3.2. Global search phase

When considering multiple conflicting objective functions, the concept of Pareto dominance relationship must be used to look for Pareto-optimal solutions [22]. A Pareto solution is one in which an improvement in one objective requires worsening another objective [21]. This means that a solution candidate $\boldsymbol{X}$ dominates another solution candidate $\boldsymbol{Y}$ for a vector-valued objective function $\boldsymbol{F}$ when $f_i(\boldsymbol{X}) \leq f_i(\boldsymbol{Y})$, for $i = 1, \ldots, m$ and, $f_j(\boldsymbol{X}) < f_j(\boldsymbol{Y})$ for some j. The set of nondominated solutions constitutes PF, and any of these solutions is optimal and provides decision-makers with a range of options to choose from based on their preferences and priorities.

In the past 20 years, a large number of metaheuristic programming methods have been proposed to solve problems with multiple conflicting objectives and to construct efficient good approximations of PF. Among these methods, two main classes seem to be more wildly used, namely evolutionary and swarm-based techniques [52]. They are simple and robust population-based search metaheuristics, attempting to find multiple Pareto-optimal solutions in a single simulation by emphasizing multiple nondominated and isolated solutions belonging to the PF. A large number of these algorithms are regrouped in various numerical optimization libraries. These algorithms have been successfully applied to a wide range of real-world problems in various fields, such as engineering, finance, and scheduling, where multiple objectives need to be considered simultaneously. Consequently, the second phase can be performed using a large diversity of metaheuristics. Hereafter, we are going to test the following techniques from PlatEMO [38, 47]:

https://doi.org/10.1017/S0263574724000766 Published online by Cambridge University Press

- NSGAII: A fast and elitist multi-objective genetic algorithm [53].
- ANSGAIII: An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach [54].
- CCMO: A coevolutionary framework for constrained MOOPs [55].
- C3M: A multistage algorithm for solving MOOPs with multi-constraints [56].

Unfortunately, the search operators used in these metaheuristic techniques are generic and do not guarantee that Pareto optimal solutions will be found in a finite number of solution evaluations for an arbitrary problem. Thus, it is suggested to postprocessing the result of this phase by using another deterministic search strategy having better convergence properties.

3.3. Local search phase

In this third phase, a deterministic algorithm is used to search for new nondominated solutions by examining the neighborhood of solutions generated in the second phase. However, a scalarization method is necessary for converting the MOOP into a single-objective optimization problem and subsequently solved it using an adequate local deterministic search algorithm. In this paper, we are going to use two methods: the goal attainment method (GAM) [57, 58] and the distance to a reference objective method (DROM) [59, 60].

GAM is a variant of goal programming method and solves the MOOP defined in rel. (7) by converting it into the following nonlinear programming problem [57]:

$$\min_{\lambda, \boldsymbol{X}} \lambda \tag{8}$$

subject to:

$$f_i(\boldsymbol{X}) - w_i\lambda \leq f_i^*,\ i = 1,\ldots, m$$
$$\boldsymbol{h}(\boldsymbol{X}) = 0, \quad \boldsymbol{g}(\boldsymbol{X}) \leq 0, \quad \boldsymbol{x_{min}} \leq \boldsymbol{X} \leq \boldsymbol{x_{max}}$$

here, f_i^*, $i = 1,\ldots, m$, represent a set of designated targets which are associated with the m objective functions $[f_1(X), f_2(X), \ldots, f_m(X)]$ corresponding to the original cost functions (6a-d). The coefficients $w_i \geq 0$, $i = 1,\ldots, m$, are weights specified by the user. The minimization of the attainment factor λ leads to finding a nondominated solution which under- or over-attains the specified goals to a degree represented by the quantities $w_i\lambda$. The MatLab function "*fgoalattain*" implements the GAM method.

DROM allows also to transform a MOOP into a mono-objective one. It is based on the minimization of a sum quantity corresponding to a distance function L_r which is defined as follows [59]:

$$L_r = \left(\sum_{i=1}^{m} \left| f_i(\boldsymbol{X}) - f_i^* \right|^r \right)^{\frac{1}{r}} \tag{9}$$

where $1 \leq r \leq \infty$ and f_i^*, $i = 1,\ldots, m$, represent a set of specified objective targets. In case $r = 2$, we deal with the Euclidean distance, whereas, $r = \infty$, the method is called the min-max method or the Tchebychev method [59]. Note that normalization and weight coefficients can be introduced in the distance formula (9) in order to better orient the search process. The MatLab function "*fminimax*" implements the DROM method.

The application of GAM or DROM leads in general to a nonlinear optimization problem that can be solved using various nonlinear mono-objective optimization techniques; hereafter, we are particularly concerned by deterministic ones in order to acquire the inherent local convergence features of these techniques. In fact, the choice for an appropriate solution algorithm is based on the inherent characteristics of the problem at hand (i.e., linear, nonlinear, differentiable, nondifferentiable, and so on). In general, the deterministic algorithms can guarantee finding at least local optimum [59, 61]. Given the same initial conditions and inputs, a deterministic algorithm, such as the sequential quadratic programming (SQP),

https://doi.org/10.1017/S0263574724000766 Published online by Cambridge University Press

Nelder–Mead, pattern search (PS), or DIRECT algorithms, will always give the same solution. These deterministic algorithms are well documented and are also available in various numerical optimization toolbox with multiple languages.

3.4. Postprocessing phase

The optimization process in the third phase aims to generate new solutions in the neighborhood of each solution of the PF approximation obtained during the global search phase. However, this local search phase might generate non-Pareto solutions and not well-distributed solutions on the entire PF. Indeed, we may lose the solutions diversity of the initial PF approximation because the local search does not try to preserve it. As a consequence, the new-generated nondominated neighbors should be compared with existing solutions, and only nondominated solutions are saved.

An easy way to do that, without losing the local convergence characteristics of the third phase, is to postprocess points obtained by the local search (phase 3) and those constituting the approximation of PF from the global search (phase 2) by using a multi-objective direct search method such as the multi-objective pattern search algorithm (MOPSA) [62].

MOPSA is a deterministic multi-objective technique based on the PS algorithm [63], which is a well-established local derivative-free optimization algorithm. MOPSA starts the search from a set of candidate solutions delivered by both the local search process and the global search process. A set of patterns is defined based on which the solutions will be updated. These patterns determine the search directions in the objective space. The algorithm performs the space exploration according to specific search and poll steps and updates solution archives until a convergence criterion is met. Theoretically, the algorithm converges to points near the true PF [62]. The MatLab function "*paretosearch"* proposes an interesting implementation of MOPSA.

4. Numerical simulations

4.1. Multi-objective trajectory planning for a planar two-dof robot

We consider in this section the case of a two-link planar manipulator executing a point-to-point task. This robot served as a benchmark for many references dealing with the minimum time TPP [9–11]. According to rel. (1), the inverse dynamic of this robot is as follows:

$$M_{11}(\boldsymbol{q})\,\ddot{q}_1 + M_{12}(\boldsymbol{q})\,\ddot{q}_2 + C_1(\boldsymbol{q},\dot{\boldsymbol{q}}) = \Gamma_1(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}) \tag{10}$$

$$M_{12}(\boldsymbol{q})\,\ddot{q}_2 + M_{22}(\boldsymbol{q})\,\ddot{q}_2 + C_2(\boldsymbol{q},\dot{\boldsymbol{q}}) = \Gamma_2(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})$$

such as

- $M_{11}(\boldsymbol{q}) = I_1 + I_2 + m_1 b_1^2 + m_2(l_1^2 + b_2^2 + 2l_1 b_2 \cos q_2) + M(l_1^2 + l_2^2 + 2l_1 l_2 \cos q_2)$
- $M_{12}(\boldsymbol{q}) = I_2 + m_2(b_2^2 + l_1 b_2 \cos q_2) + M(l_2^2 + l_1 l_2 \cos q_2)$
- $M_{22}(\boldsymbol{q}) = I_2 + m_2 b_2^2 + M l_2^2$
- $C_1(\boldsymbol{q},\dot{\boldsymbol{q}}) = -l_1 \dot{q}_2(2\dot{q}_1 + \dot{q}_2)(m_2 b_2 \sin q_1 + M l_2 \sin q_2)$
- $C_2(\boldsymbol{q},\dot{\boldsymbol{q}}) = l_1 \dot{q}_1^2(m_2 b_2 \sin q_1 + M l_2 \sin q_2)$

The associated data are reported in Table 1, and two scenarios are discussed for this robot. For both cases, the robot is required to move from an initial configuration $(q_{1i}, q_{2i}) = (0, 0)$ to the final configuration $(q_{1f}, q_{2f}) = (1, -0.5)rad$ with null limit velocities. Also, we are interested in solving the MOOP for a TPP involving minimizing simultaneously F_1, the transfer time (rel. (6a)), and F_2, the mean value of applied joint torques (rel. (6c)). The motion is to plan under constraints on joint speed and torques amplitudes. Two scenarios are treated: the first one involves transfer in free space and the second one corresponds to a transfer in the presence of one obstacle.

https://doi.org/10.1017/S0263574724000766 Published online by Cambridge University Press

Table I. *Data of a two-dof planar robot.*

l_1=2b_1 [m]	l_2=2b_2 [m]	m_1[kg]	m_2 [kg]	M [kg]	I_1 [kg.m^2]
0.4	0.25	29.58	15	6	0.417
$T^{\max}$ [s]	$\dot{q}_1^{max}$ [rad/s]	$\dot{q}_2^{max}$ [rad/s]	Γ_1^{max} [N.m]	Γ_2^{max}[N.m]	I_2[kg.m^2]
5	3	8	25	9	0.206

As an application of the first phase of the proposed approach, the time evolution of the joint position variable ($q_1(t)$, $q_2(t)$) is approximated using clamped cubic spline functions interpolating a set of two uniformly distributed control points ($N_p = 2$). A superior-born T_{max} of five seconds is fixed for the transfer time.

Optimization techniques are exploited from PlatEMO and the Matlab optimization toolbox. In particular, four different population-based methods (NSGAII, ANSGAIII, CCMO, and C3M) have been selected from PlatEMO [38, 47] to be used in the second phase. For these techniques, the optimization process has been performed using the following parameters: population size = 100, number of generations = 50, and the maximum number of available function evaluations which is thus fixed to 5000. For the achievement of the local search of phase 3, both "*fgoalattain*" and "*fminimax*" MatLab functions are used to perform, respectively, GAM- and DROM-based optimizations. Additionally, the final phase is executed using the "*paretosearch*" MatLab function. All these MatLab functions are executed using their default parameters. It is important to recall that the various optimization techniques applied in phases 2 and 3 are presented in order to illustrate the flexibility of the proposed framework and not for comparison between them[1].

Mainly two metrics are used to evaluate the quality of obtained PF at the end of phases 2 and 4, namely hypervolume indicator "HI" and average distance "AD." HI measures the size of the dominated space corresponding to the total volume of polytope defined by Pareto points in the objective function space. The closer points move to the PF, and the more they distribute along PF, the more space gets dominated; thus, HI increases [25]. AD is a measure of the closeness of an individual of the PF to its nearest neighbors, and it measures distance among individuals of the same rank in objective function space, so low values of AD are better.

4.1.1 Scenario 1: Free point-to-point trajectory planning task

As a first scenario, the robot is moving in a space free of obstacles. Figures (3), (4a-d), and (5) illustrate the solutions in the objective function space obtained at the end of phases 2, 3, and 4. Table II regroups the main performances recorded at the end of phases 2 and 4. The analyze of these results shows that:

- The four techniques applied during the second phase delivered different approximations of the PF with different metrics as indicated in Table II and as it is noticed in Fig. 3.
- Figure 4 illustrate the fact that the local search of phase (3) based on GAM or DROM was able to enhance the majority of solutions obtained in phase (2). However, sometimes the optimization process delivers dominated solutions that should be filtered later. Also, we noticed that each point of the approximated PF of the second phase involved about two seconds to be treated using GAM or DROM. The duration of phase 2 depends on the size of PF obtained at the end of phase 1.

[1] Simulations are performed on a PC with Intel processor CORE I3-6006U CPU 2 GHz with 8GB RAM, using Matlab® 2021a, under Windows10.

https://doi.org/10.1017/S0263574724000766 Published online by Cambridge University Press

Table II. *Main performances recorded at the end of phases 2 and 4.*

Technique	CPU Time [s]	HI	AD	Min (F_1)	Max (F_1)	Min (F_2)	Max (F_2)
Phase 2							
NSGAII	45.31	7.8542	0.0091	1.2744	5	0.0116	0.7563
CCMO	35.64	7.5390	0.0146	1.3125	5	0.0117	0.7010
ANSGAIII	30.86	7.1687	0.0141	1.3324	5	0.0115	0.6206
C3M	43.75	9.4483	0.0274	1.1668	5	0.0115	1.0730
Phase 4							
NSGAII-PS	68.98	8.2517	0.0046	1.2106	5	0.0114	0.8220
CCMO-PS	61.91	7.9975	0.0058	1.2322	5	0.0114	0.7736
ANSGAIII-PS	61.84	7.7355	0.0050	1.2572	5	0.0114	0.7233
C3M-PS	65.53	9.4674	0.0072	1.1277	5	0.0114	1.0574

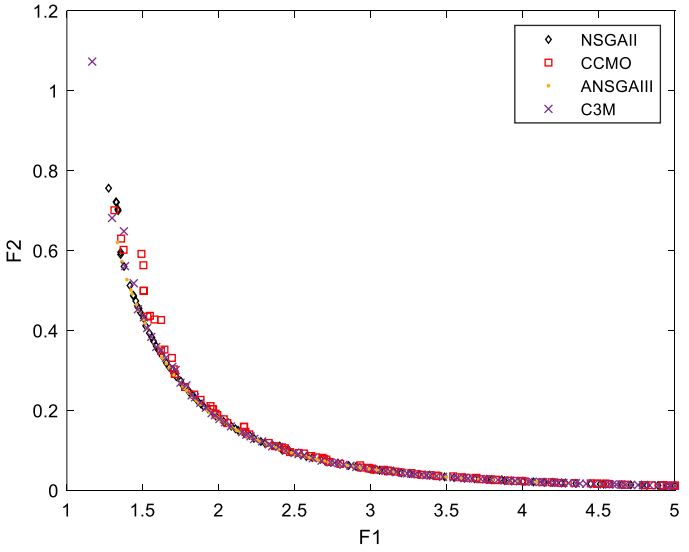
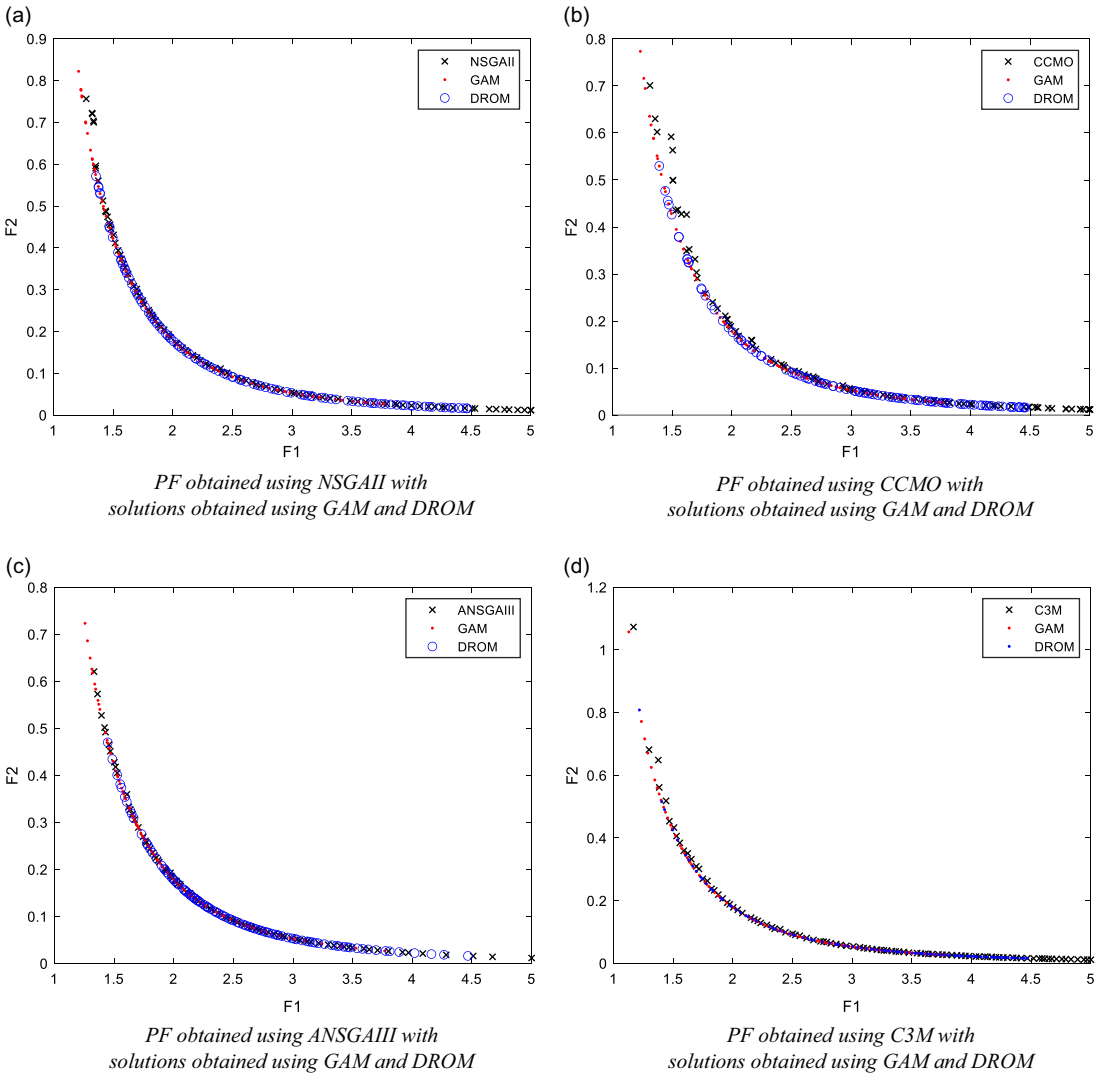


Figure 3. *PF obtained at the end of phase 2 using four different population-based metaheuristics.*

- Additionally, as illustrated in Fig. 5, the fourth phase was able to eliminate dominated solutions and enhance the diversity of the final constructed PF. Also, although the applied optimization techniques during the second phase delivered different approximations of the PF, these differences disappeared in the final PF approximations obtained at the end of the fourth phase; they all converge to the same area in the objective function space. The difference resides in the covered range of the PF as indicated in Table II by the min/max values of the objective functions F_1 and F_2.
- Figure 6 illustrates one selected solution from PF obtained using C3M-PS for which the transfer time is $F_1 = 1.1092$s, and the normalized mean square value of applied torques is about $F_2 = 1.1371$s. As an indication, the minimum time solution under dynamic constraint obtained in ref. [11] using a nonlinear optimal control approach is bang-bang in terms of actuator torques and the transfer is achieved in a duration $F_1 = 1.002$ S (decrease of about 10%), but the corresponding *normalized* mean square value of applied torques is about $F_2 = 2.02$s (increase of about 90%).

https://doi.org/10.1017/S0263574724000766 Published online by Cambridge University Press



PF obtained using NSGAII with solutions obtained using GAM and DROM

PF obtained using CCMO with solutions obtained using GAM and DROM

PF obtained using ANSGAIII with solutions obtained using GAM and DROM

PF obtained using C3M with solutions obtained using GAM and DROM

Figure 4. *Results of local search versus approximations of PF obtained by the global search.*

4.1.2 Scenario 2: point-to-point trajectory planning in the presence of one obstacle

For this scenario, the robot is required to link the same extreme configurations of the first scenario, but this time, in the presence of one circular obstacle of radius $0.1m$ located at the point of coordinates ($x = 0.45$m, $y = 0.25$m).

This scenario is treated exclusively using the combination of optimization techniques C3M-DROM-MOPSA but using different number of free control nodes. The optimizations are conducted using the same parameters as in the previous scenario. Table III regroups some characteristic values of the obtained PFs, and Fig. 7 illustrates the results of different phases in the objectives space. We can see that the number of free control points influences the quality of the constructed PF and the computing efforts. Low AD values are obtained for low number (2 and 3) of free nodes, whereas high HI values are obtained using relatively high number (4 and 5) of free nodes. This is the consequence of modifying the size of the search space: increasing the number of free nodes makes the joint trajectory model more flexible and enables us to represent a large spectrum of solutions but, in the same time, augments the search space and makes its exploration more difficult and time-consuming.

https://doi.org/10.1017/S0263574724000766 Published online by Cambridge University Press

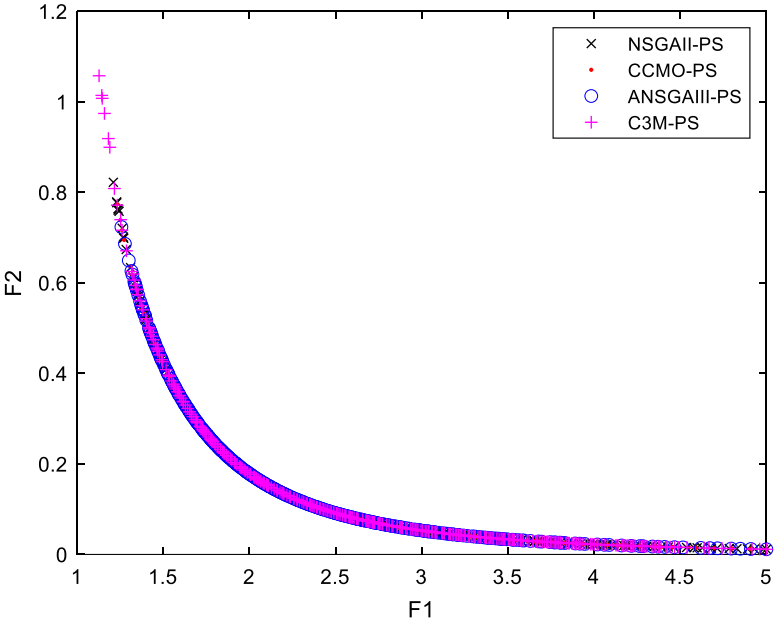


Figure 5. *Superposition of Pareto fronts obtained using various combinations of optimization techniques at the end of phase 4.*

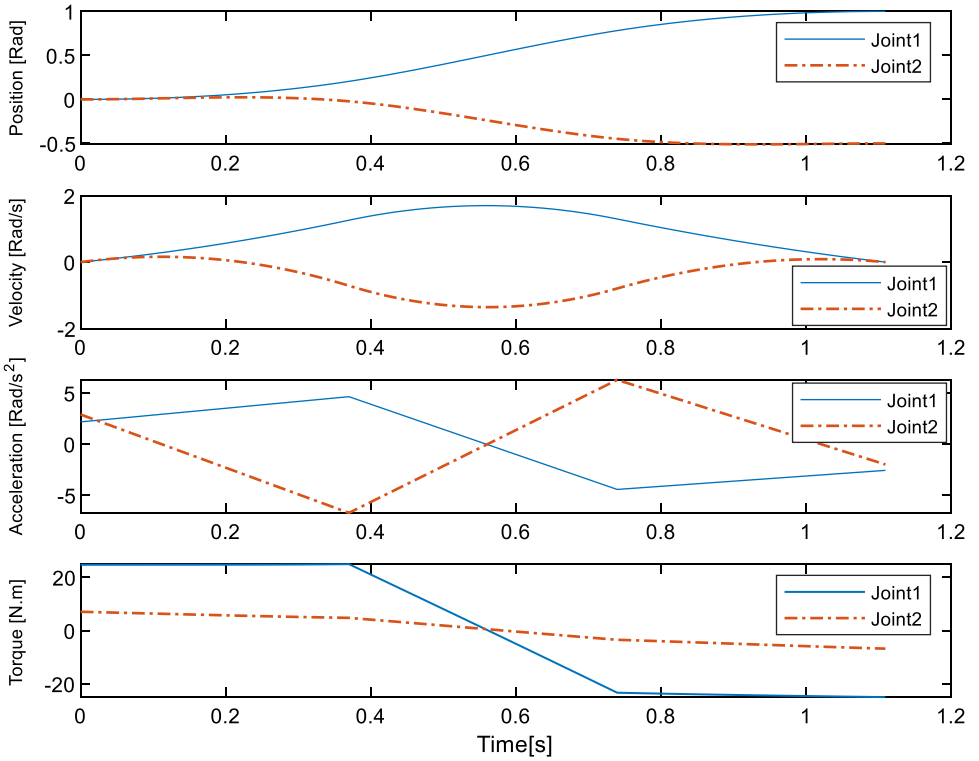


Figure 6. *Joint's trajectory corresponding to the minimum time solution selected from the PF obtained using the combination C3M-DROM-MOPSA.*

https://doi.org/10.1017/S0263574724000766 Published online by Cambridge University Press

Table III. *Main performances recorded at the end of phase 4 in the presence of one obstacle.*

Number of free control nodes	**2**	**3**	**4**	**5**
Min(F1)	1.663	1.485	1.398	1.480
Max (F1)	5	5	5	5
Min(F2)	0.028	0.024	0.023	0.022
Max (F2)	0.769	0.938	1.347	1.135
HI	7.062	8.088	10.021	9.570
AD	0.012	0.014	0.032	0.028
CPU time	625	621	1008	1117

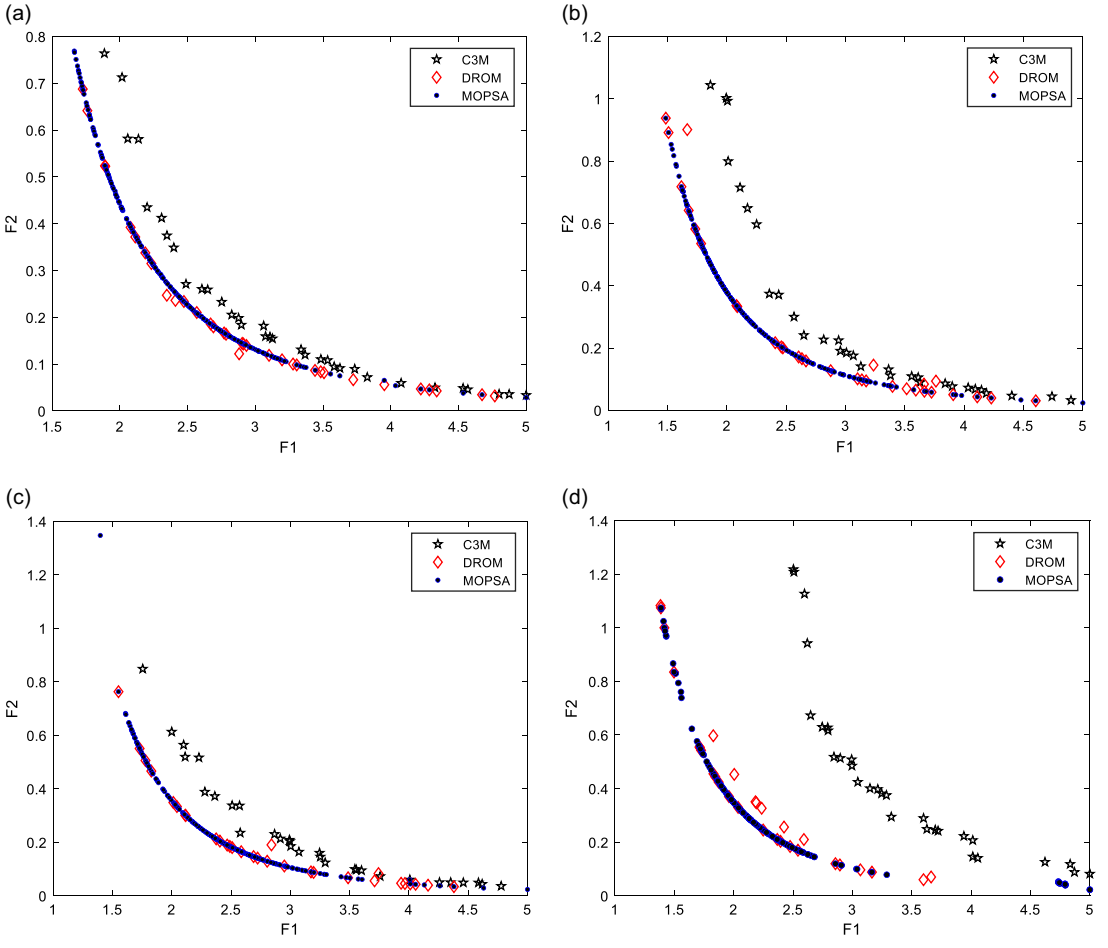


Figure 7. *Results of phases 2, 3, and 4 using various number of free control points: (a) 2 points, (b) 3 points, (c) 4 points, and (d) 5 points.*

Globally, versus the previous scenario, the motion of the robot becomes slower and involves more efforts. This is due to the presence of the obstacle. Indeed, joining the same extreme configurations involves maneuvers in order to avoid collisions. As an illustration, we have presented in Fig. 8 the fastest transfer recorded in the PF obtained using four nodes. We can see that all imposed constraints

https://doi.org/10.1017/S0263574724000766 Published online by Cambridge University Press

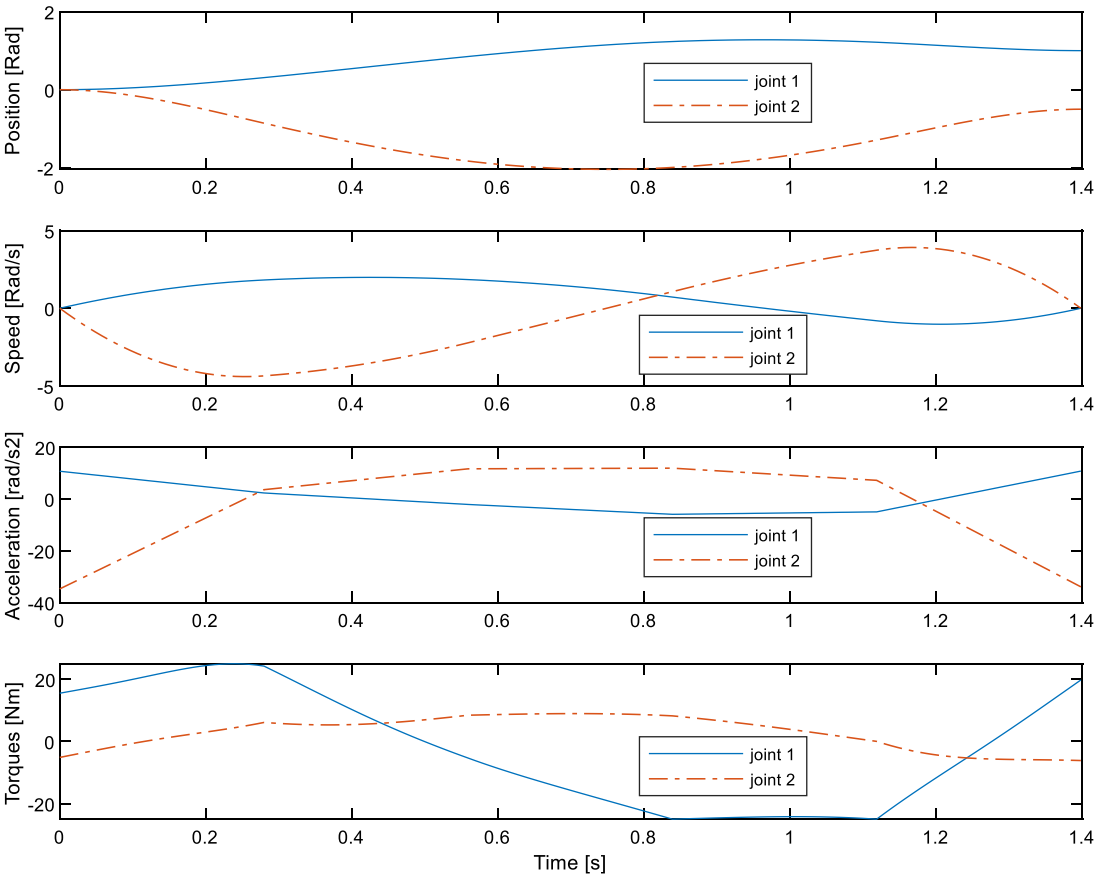


Figure 8. *Time evolution of joint trajectories corresponding to minimum time solution obtained using four nodes.*

are respected and the motion is executed in a smooth way. Figure 9b illustrates the corresponding robot motion in the presence of the obstacle, and it can be compared with the previous scenario 1 (Fig. 9a).

4.2. Multi-objective trajectory planning for a six-dof robot

In this case, the goal is to plan optimal time–jerk trajectories for the PUMA 560 robot [64] and to compare the results with those given in Ref. [31, 35]. The manipulator is required to move through a set of via points which are reported in Table IV while respecting limited kinematic performances reported in Table V.

Huang *et al.* [31] adopted the fifth-order B-spline interpolation method to construct the trajectory in the joint space with null limit conditions on velocity, acceleration, and jerk. The vector of optimization variables includes only time intervals ($h_i = t_i - t_{i-1}$) defining the horizontal positions of the via points along the timescale. Two objective functions are considered: the transfer time (*rel.* (*6a*)) and the global mean jerk which is a modified expression of rel. (6b). Thus, the vector $\boldsymbol{F_{obj}}$ to be minimized is as follows:

$$\boldsymbol{F_{obj}} = \begin{bmatrix} T = \sum_{i=1}^{m-1} h_i \\ JM = \sum_{j=1}^{n} \sqrt{\frac{1}{T}\int_0^T \dddot{q}_j^2\,(t)\,dt} \end{bmatrix}, (n = 6, m = 4) \tag{11}$$

https://doi.org/10.1017/S0263574724000766 Published online by Cambridge University Press

Table IV. *Knot angles of the via points defining the transfer task.*

	***Joint* [*deg*]**					
Knot	**1**	**2**	**3**	**4**	**5**	**6**
1	−10	20	15	150	30	120
2			***Virtual***			
3	60	50	100	100	110	60
4	20	120	−10	40	90	100
5			***Virtual***			
6	55	35	30	10	70	25

Table V. *Kinematic constraints.*

	Joint					
Constraints	**1**	**2**	**3**	**4**	**5**	**6**
Velocity $\dot{q}_j^{max}$ *(degrees/s)*	100	95	100	150	130	110
Acceleration $\ddot{q}_j^{max}$ *(degrees/s*^{2}*)*	60	60	75	70	90	80
Jerk $\dddot{q}_j^{max}$*(degrees/s*^{3}*)*	60	66	85	70	75	70

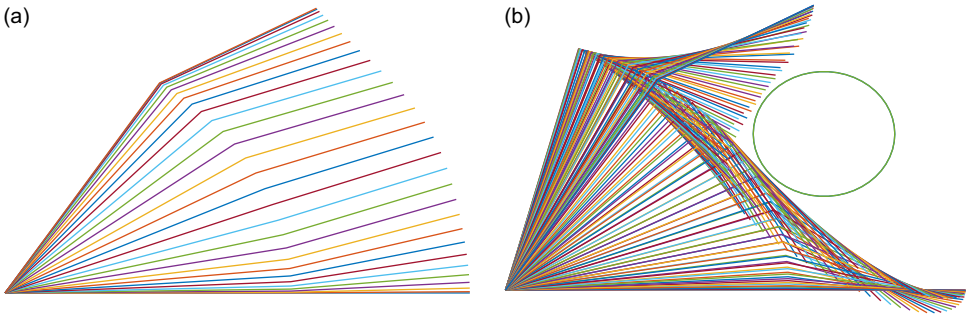


Figure 9. *Stroboscopic robot motion corresponding to the minimum time solution: (a) scenario 1 and (b) scenario 2.*

This problem is treated according to the proposed approach as follows. In the first phase, a fifth-order spline function interpolation technique is used to construct joint trajectories taking into account limit conditions and via points angles of Table IV. The transfer duration T is limited to 25 s. In the second phase, the C3M technique is applied using the same parameters of the previous example. In the third phase, the local search is applied according to the DROM approach using the "*fminimax*" MatLab function. The final phase is executed using the "*paretosearch*" MatLab function. The obtained PF is illustrated in Fig. 10. The range of the final PF is [8.496s, 25s] × [8.168°/s^3, 207°/s^3]. In ref. [31], the same problem is solved by using NSGA-II, and the obtained PF range is [9.058s, 13.96s]×[55.55°/s^3, 188.98 °/s^3]. Also in ref. [35], this example has been treated using the constrained multi-objective particle swarm algorithm, and the range of obtained PF is [8.722s,15.072s] × [29.26°/s^3 153.41°s^3]. Of course, the shortest execution time corresponds to the largest jerk index and vice versa. The quality of our final PF is better, and the minimum time solution found using our approach is the best one, and it is illustrated in Fig. 11. We can see that generated trajectories are smooth in terms of position, velocity, acceleration, and jerk for all joints. Also, boundary and intermediate conditions are respected.

https://doi.org/10.1017/S0263574724000766 Published online by Cambridge University Press

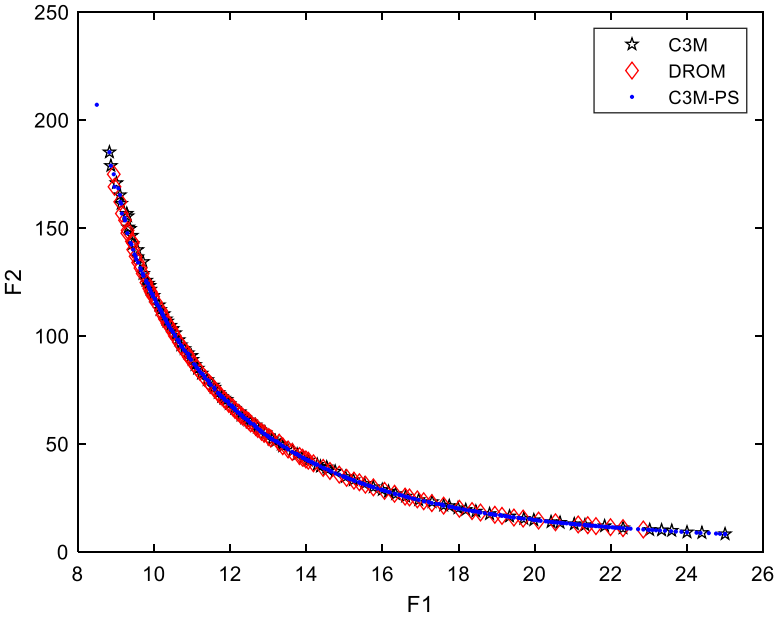


Figure 10. *Pareto front for time–jerk optimal trajectory planning problem.*

5. Conclusion

Multi-objective trajectory planning is a powerful tool that can be used to improve the performance of industrial robots. By optimizing multiple objectives simultaneously, it can help to increase productivity, reduce energy consumption, and improve the quality. In fact, there is no single best solution to a MOOP, as the different objectives may conflict with each other. Instead, the goal is to find a set of compromise solutions constituting the Pareto-optimal front.

The recent development of powerful multi-objective optimization techniques and their availability to the research community in various digital libraries encouraged us to propose in this paper a hybrid optimization framework to generate optimal reference trajectories for robotic manipulators. These trajectories are compromise solutions of the intricate original multi-objective optimal control problem. This framework includes four phases. The first one is a transcription phase where an adequate parametrization scheme is adopted for the robot configuration variables. In the second phase, a population-based search technique is used to globally explore the solution space and reach the region near the targeted optimal PF in a relatively small number of generations. Then, in the third phase, a deterministic optimization technique is used to search iteratively for new nondominated points by examining the neighborhood of solutions obtained in the second phase. Finally, solutions obtained in the second and the third phases are gathered in a unique population and postprocessed in the fourth phase in order to remove possible non-Pareto optimal points and to enhance the quality of the final PF.

The proposed framework is flexible and has been implemented according to a high-level relay hybrid model using different optimization techniques. It has a robust search mechanism thanks to the population-based component, and it is having the theoretical properties of convergence of the deterministic component. It has been successfully applied to solve problems involving two- and six-*dof* robots under various kinodynamic constraints and optimizing various cost functions. These results have demonstrated the efficiency of our proposal and opened the door for further applications. Indeed, based on this work, other studies can be envisaged in the future. For example, the influence of the discretization scheme parameters on the final PF can be investigated in deep. Also, instead of using only one metaheuristic, the global search phase can be also performed using a combination of various population-based metaheuristics, and this idea can also be tested. As well, it is interesting to study the possibility of introducing

https://doi.org/10.1017/S0263574724000766 Published online by Cambridge University Press

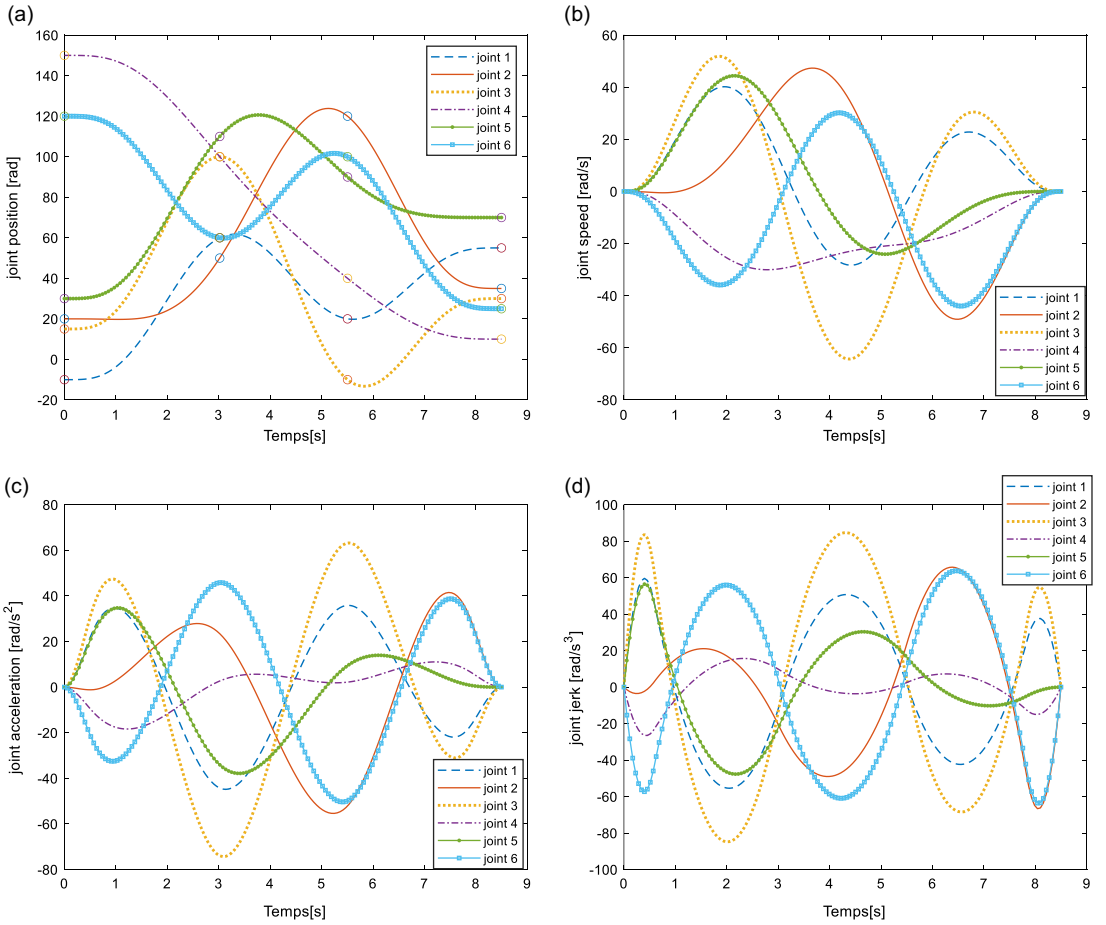


Figure 11. *Joint's trajectory for the PUMA 560 robot corresponding to the minimum time solution selected from the PF obtained using the combination C3M-DROM-MOPSA: (a) position (b) velocity, (c) acceleration, and (d) Jerk.*

clustering techniques in order to reduce the number of solutions proposed to the decision-maker for selecting the best compromise solution. Finally, extending the use of the proposed approach for other classes of robots (e.g., mobile robots and aerial robots) taking into account their particularities is an interesting perspective.

Authors contributions. All the work is done by Taha CHETTIBI.

Financial support. This research received no specific grant from any funding agency.

Competing interests. The author declares no competing interests exist.

Ethical considerations. The submitted work is original and not has been published elsewhere in any form or language.

References

[1] J. J. Craig. *Introduction to Robotics: Mechanics and Control* 4th edn, (Pearson Education Limited, 2022).
[2] M. Kahn, The near-minimum time control of open-loop articulated kinematic linkages Tech. Rep. AIM-106, (1969). Stanford University.

https://doi.org/10.1017/S0263574724000766 Published online by Cambridge University Press

[3] M. E. Kahn and B. Roth, "The near minimum time control of open loop articulated kinematic chains," *ASME J Dyn Sys Meas Cont* **11**(3), 164–172 (1971).

[4] J. E. Bobrow, S. Dubowsky and J. S. Gibson, "Time-optimal control of robotic manipulators," *Int J Robot Res* **4**(3), 3–17 (1985).

[5] J. T. Betts, "Survey of numerical methods for trajectory optimization," *J Guid Cont Dyn* **21**(2), 193–207 (1998).

[6] A. C. Bruce, "A survey of methods available for the numerical optimization of continuous dynamic systems," *J Optim Theory Appl* **152**, 271–306 (2012).

[7] O. von Stryk and R. Bulirsch, "Direct and indirect methods for trajectory optimization," *Ann Oper Res* **37**(1), 357–373 (1993).

[8] O. V. Stryk and M. Schlemmer, "Optimal Control of the Industrial Robot Manutec r3," **In:** *Computational Optimal Control International Series of Numerical Mathematics* (R. Bulirshn and D. Kraft, eds.) (Basel: Birkhäuser, 1994) pp. 367–382.

[9] C. R. Fotouhi and W. Szyszkowski, "An algorithm for time-optimal control problems," *J Dyn Syst Meas Cont* **120**(3), 414–418 (1998). doi: 10.1115/1.2805419

[10] M. H. Ghasemi, N. Kashiri and M. Dardel, "Time-optimal trajectory planning of robot manipulators in point-to-point motion using an indirect method," *Proceed Inst Mech Eng Part C: J Mech Eng Sci* **226**(2), 473–484 (2012).https://doi:10

[11] M. Massaro, S. Lovato, M. Bottin and G. Rosati, "An optimal control approach to the minimum-time trajectory planning of robotic manipulators," *Robotics* **12**(3), 64 (2023). doi: 10.3390/robotics12030064.

[12] T. Chettibi, H. E. Lehtihet, M. Haddad and S. Hanchi, "Minimum cost trajectory planning for industrial robots," *European J Mech-A/Soli* **23**(4), 703–715 (2004).

[13] T. Chettibi and H. E. Lehtihet, "A New Approach for Point-to-Point Optimal Motion Planning Problems of Robotic Manipulators," **In:** *Proc. of 6th Biennial Conf. on Engineering System Design and Analysis, APM10*, (2002).

[14] T. Chettibi, M. Haddad, S. Rebai and A. Hentout, "A Stochastic off Line Planner of Optimal Dynamic Motions for Robotic Manipulators," **In:** *Proceedings of 1st International Conference on Informatics, in Control, Automation and Robotics*, (2004).

[15] A. S. Rana and A. M. S. Zalazala, "Near time optimal collision free motion planning of robotic manipulators using an evolutionary algorithm," *Robotica* **14**, 621–632 (1996).

[16] J. C. Latombe. *Robot Motion Planning* (Kluwer Academic Publishers, 1991).

[17] J. C. Latombe, "Motion planning: A journey of, molecules, digital actors and other artefacts," *Int J Robot Res* **18**(11), 1119–1128 (1999).

[18] A. Gasparetto, P. Boscariol, A. Lanzutti and R. Vidoni, "Path Planning and Trajectory Planning Algorithms: A General Overview,Path Planning and Trajectory Planning Algorithms: A General Overview," **In:** *Motion and Operation Planning of Robotic Systems*, (Springer, 2015) pp. 3–27.

[19] R. T. Marler and J. S. Arora, "Survey of multi-objective optimization methods for engineering," *struct Multidisc Optim* **26**(6), 369–395 (2004). doi: 10.1007/s00158-003-0368-6.

[20] K. Deb, "Multi-Objective Optimisation Using Evolutionary Algorithms: An Introduction," **In:** *Multi-Objective Evolutionary Optimisation for Product Design and Manufacturing*, (Springer Verlag, (2011). doi: 10.1007/978-0-85729-652-8_1

[21] J. L. J. Pereira, G. A. Oliver, M. B. Francisco, S. S. Cunha and G. F. Gomes, "A review of multi-objective optimization: Methods and algorithms in mechanical engineering problems," *Arch Comput Method E* **29**(4), 2285–2308 (2022). doi: 10.1007/s11831-021-09663-x.

[22] F. Logist, B. Houska, M. Diehl and J. Van Impe, "Fast pareto set generation for nonlinear optimal control problems with multiple objectives," *Struct Multidisc Optim* **42**(4), 591–603 (2010). doi: 10.1007/s00158-010-0506-x

[23] A. Gasparetto and V. Zanotto, "Optimal trajectory planning for industrial robots," *Adv Eng Softw* **41**(4), 548–556 (2010). doi: 10.1016/j.advengsoft.2009.11.001.

[24] X.-S. Yang. *Optimization techniques and applications with examples* (JohnWiley & Sons Inc., 2018). ISBN: 9781119490609.

[25] M. T. M. Emmerich and A. H. Deutz, "A tutorial on multiobjective optimization: Fundamentals and evolutionary methods," *Nat Comput* **17**(3), 585–609 (2018). doi: 10.1007/s11047-018-9685-y

[26] R. Saravanan, S. Ramabalan and C. Balamurugan, "Evolutionary optimal trajectory planning for industrial robot with payload constraints," *Int J Adv Manuf Technol* **38**, 1213–1226 (2008). doi: 10.1007/s00170-007-

[27] R. Saravanan, S. Ramabalan and C. Balamurugan, "Evolutionary multi-criteria trajectory modeling of industrial robots in the presence of obstacles," *Eng Appl Artif Intell* **22**, 329–342 (2009). doi: 10.1016/j.engappai.20

[28] Z. Xu, S. Li, Q. Chen and B. Hou. MOPSO Based Multi-Objective Trajectory Planning for Robot Manipulators. **In:** *2nd International Conference on Information Science and Control Engineering*, (2015) pp. 824–828.

[29] X. Shi, H. Fang and L. Guo, "Multi-Objective Optimal Trajectory Planning of Manipulators Based on Quintic NURBS," **In:** *2016 IEEE International Conference on Mechatronics and Automation*, (2016) pp. 759–765. doi:, 10.1109/ICMA.2016.7558658

[30] F. J. Abu-Dakka, F. J. Valero, J. L. Suner and V. Mata, "A direct approach to solving trajectory planning problems using genetic algorithms with dynamics considerations in complex environments," *Robotica* **33**(3), 669–683 (2018).

[31] J. Huang, P. Hu, K. Wu and M. Zeng, "Optimal time-jerk trajectory planning for industrial robots," *Mech Mach Theory* **121**, 530–544 (2018).

[32] A. Rout, G. B. Mahanta, D. Bbvl and B. B. Biswal, "Kinematic and dynamic optimal trajectory planning of industrial robot using improved multi-objective ant lion optimizer," *J Inst Eng India Ser:C* **101**(3), 559–569 (2020).

[33] J. Lan, Y. Xie, G. Liu and M. Cao, "A multi-objective trajectory planning method for collaborative robot," *Electronics* **9**(5), 859 (2020). doi: 10.3390/electronics9050859.

[34] G. Wu, W. Zhao and X. Zhang, "Optimum time-energy-jerk trajectory planning for serial robotic manipulators by reparameterized quintic NURBS curves," *Proceed Inst Mech Eng Part C: J Mech Eng Sci* **235**(19), 4382–4393 (2021). doi: 10.1177/0954406220969734.
[35] X. Zhang and G. Shi, "Multi-Objective optimal trajectory planning for manipulators in the presence of obstacles," *Robotica* **40**(4), 888–906 (2022). doi: 10.1017/S0263574721000886 Cambridge University Press
[36] Q. Cheng, X. Hao, Y. Wang, W. Xu and S. Li, "Trajectory planning of transcranial magnetic stimulation manipulator based on time-safety collision optimization," *Robot Auton Syst* **152**, 104039 (2022).
[37] Q. Shi, Z. Wang, X. Ke, Z. Zheng, Z. Zhou, Z. Wang, Y. Fan, B. Lei and P. Wu, "Trajectory optimization of wall-building robots using response surface and non-dominated sorting genetic algorithm III," *Automat Constr* **155**, 105035 (2023). doi: 10.1016/j.autcon.2023.105035.
[38] Y. Tian, R. Cheng, X. Zhang and Y. Jin, "PlatEMO: A MATLAB platform for evolutionary multi-objective optimization," *IEEE Comput Intell Mag* **12**(4), 73–87 (2017). doi: 10.1109/MCI.2017.2742868.
[39] J. Blank and K. Deb, "Pymoo: multi-objective optimization in python, "*IEEE Access* **8**, 89497–89509 (2020). doi: 10.1109/ACCESS.2020.2990567.
[40] C. Coello, "Recent Trends in Evolutionary Multiobjective Optimization," **In:** *Evolutionary Multiobjective Optimization*, A. Abraham, L. Jain and Goldberg R.eds.) (Springer, 2005) pp. 7–32.
[41] E. Zitzler, K. Deb and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evol Comput* **8**(2), 173–195 (2000).
[42] K. Sindhya, K. Miettinen and K. Deb, "A hybrid framework for evolutionary multi-objective optimization," *IEEE Trans Evolu Comput* **17**(4), 495–511 (2013).
[43] E.-G. Talbi, "Hybrid metaheuristics for multi-objective optimization," *J Algor Comp Technol* **9**(1), 41–63 (2015).
[44] P. Alotto and G. Capasso, "A deterministic multiobjective optimizer," *COMPEL-J Comput Math Electr Electron Eng* **34**(5), 1351–1363 (2015). doi: 10.1108/COMPEL-03-2015-0117.
[45] The Mathworks, Inc, USA, MATLAB Global Optimization Toolbox User's Guide, (2020).
[46] R. Berger, M. Bruns, A. Ehrmann, A. Haldar, J. Hafele, B. Hofmeister, C. Hübler and R. Rolfes, "EngiO – object-oriented framework for engineering optimization," *Adv Eng Softw* **153**, 102959 (2021). doi: 10.1016/j.advengsoft.2020.102959.
[47] Y. Tian, W. Zhu, X. Zhang and Y. Jin, "A practical tutorial on solving optimization problems via PlatEMO," *Neurocomputing* **518**, 190–205 (2023). doi: 10.1016/j.neucom.2022.10.075.
[48] A. Jaszkiewicz, "Genetic local search for multi-objective combinatorial optimization," *European J Oper Res* **137**(1), 50–71 (2002).
[49] K. Sindhya, K. Deb and K. Miettinen, "A Local Search Based Evolutionary Multi-Objective Approach for Fast and Accurate Convergence," **In:** *Proc. 10th Parallel Problem Solving From Nature*, (2008) pp. 815–824.
[50] C. De Boor. *A Practical Guide to Splines* (Springer-Verlag, 1978).
[51] C. De Boor, Spline Toolbox for use with MATLAB, The Math Works Inc, (2005).
[52] S. Sharma and V. Kumar, "A comprehensive review on multi-objective optimization techniques: Past, present and future," *Arch Comput Method E* **29**(7), 5605–5633 (2022). doi: 10.1007/s11831-022-09778-9.
[53] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans Evolu Comput* **6**(2), 182–197 (2002). doi: 10.1109/4235.996017.
[54] H. Jain and K. Deb, "An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part II: Handling constraints and extending to an adaptive approach," *IEEE Trans Evolu Comput* **18**(4), 602–622 (2014). doi: 10.1109/TEVC.2013.2281534.
[55] Y. Tian, T. Zhang, J. Xiao, X. Zhang and Y. Jin, "A coevolutionary framework for constrained multiobjective optimization problems," *IEEE Trans Evolu Comput* **25**(1), 102–116 (2021). doi: 10.1109/TEVC.2020.3004012.
[56] R. Sun, J. Zou, Y. Liu, S. Yang and J. Zheng, "A multistage algorithm for solving multi-objective optimization problems with multi-constraints," *IEEE Trans Evolut Comput* **27**(5), 1207–1219 (2023). doi: 10.1109/TEVC.2022.3224600.
[57] S. S. Rao, "Engineering Optimization: Theory and Practice, 4th edn. (John Wiley & Sons, Inc., 2009).
[58] A. Messac. *Optimization in Practice with MATLAB® for Engineering Students and Professionals* (Cambrdige University Press, 2015).
[59] Y. Collette and P. Siarry. *Principles and Case Studies* (Springer-Verlag, 2004).
[60] G. Eichfelder. *Adaptive Scalarization Methods in Multiobjective Optimization* (Springer-Verlag, 2008).
[61] J. R. R. A. Martins and A. Ning. *Engineering Design Optimization* (Cambrdige University Press, 2021).
[62] A. L. Custòdio, J. F. A. Madeira, A. I. F. Vaz and L. N. Vicente, "Direct multisearch for multiobjective optimization," *SIAM J Optim* **21**(3), 1109–1140 (2011). doi: 10.1137/10079731X.
[63] R. Hooke and T. A. Jeeves, "" Direct Search" solution of numerical and statistical problems," *J ACM* **8**(2), 212–229 (1961). doi: 10.1145/321062.321069.
[64] P. Corke, Roboitics Vision and Control Fundamental Algorithms in MATLAB (Springer Tracts in Advanced Robotics 2nd edn, Springer, 2017).

Cite this article: T. Chettibi (2024). "Multi-objective trajectory planning for industrial robots using a hybrid optimization approach", Robotica **42**, 2026–2045. https://doi.org/10.1017/S0263574724000766