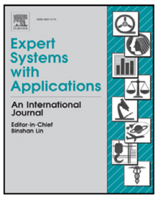
Expert Systems With Applications 136 (2019) 133–144

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa



Self-adaptive MRPBIL-DE for 6D robot multiobjective trajectory planning

Sujin Bureerat[a], Nantiwat Pholdee[a,*], Thana Radpukdee[b], Papot Jaroenapibal[b]

[a] Sustainable and Infrastructure Research and Development Center, Department of Mechanical Engineering, Faculty of Engineering, Khon Kaen University, 40002, Thailand
[b] Department of Industrial Engineering, Faculty of Engineering, Khon Kaen University, 40002, Thailand

ARTICLE INFO

Article history:
Received 18 September 2018
Revised 29 April 2019
Accepted 16 June 2019
Available online 18 June 2019

Keywords:
Multiobjective meta-heuristic algorithm
Robot trajectory planning multiobjective
Optimisation
Self-adaptive algorithm
Time-jerk minimisation

ABSTRACT

This work presents self-adaptive multiobjective real-code population-based incremental learning hybridised with differential evolution (MRPBIL-DE) for solving a 6D robot trajectory planning multiobjective optimisation problem. The objective functions are assigned to minimise travelling time and minimise maximum jerk taking place during motion while the constraints are velocity, acceleration and jerk constraints. A five order polynomial function is used to represent a motion equation while the motion path is divided into two sub-paths; from initial to intermediate positions and from intermediate to final positions. The optimiser is used to find a set of design variables including joint positions, velocities and accelerations at intermediate positions, moving time from the initial to intermediate positions, and that from the intermediate to final positions. Several multiobjective meta-heuristics (MOMHs) along with the proposed algorithm are used to solve the trajectory optimisation problem of robot manipulators while their performances are investigated. The results indicated that the proposed algorithm is effective and efficient for multiobjective robot trajectory planning optimisation problem. The results obtained from such a method are set as the baseline for further study of robot trajectory planning optimisation.

1. Introduction

Meta-heuristics (MHs), while some of them are classified as evolutionary algorithms (EAs), are popularly used for engineering design activities in various fields due to the advantages of its derivative-free feature. Therefore, they can deal with almost any kind of design variables, objective and constraint functions. Also, the multiobjective versions of MHs can explore Pareto fronts within a single run. However, the MHs also have some inevitable disadvantages i.e. slow convergence rate and a lack of search consistency particularly for multiobjective problems. This leads to the development of numerous new MHs over the last decade aiming at high search performance.

Robot trajectory planning is an optimisation problem in which MHs can be applied as optimisers. The objective functions can be minimisation of travelling time (Tangpattanakul & Artrit, 2009), minimising energy (Garg & Kumar, 2002) or minimising jerk (Piazzi & Visioli, 2000) while constraints include positions, velocities and accelerations. MHs that have been successfully used in dealing with a single objective problem of the trajectory planning were, for example, harmony search algorithm (HS) (Tangpattanakul & Artrit, 2009), artificial bee colony algorithm (ABC) (Mac, Copot, Tran, & De Keyser, 2016; Savsani, Jhala, & Savsani, 2013; Wichapong, Bureerat, & Pholdee, 2018), particle swarm optimization (PSO) (Mac et al., 2016; Wichapong et al., 2018), teaching learning based optimization (TLBO) (Mac et al., 2016; Savsani et al., 2013; Wichapong et al., 2018), etc. Multi-objective meta-heuristics (MOMHs) successfully used for trajectory planning, on the other hand, were a multiobjective genetic algorithm (MOGA) (Pires, de Moura Oliveira, & Machado, 2007), a non-dominated sorting genetic algorithm (NSGA-II) (Deb, Pratap, Agarwal, & Meyarivan, 2002; Huang, Hu, Wu, & Zeng, 2018) and multiobjective particle swarm optimisation (MOPSO) (Mac, Copot, Tran, & Keyser, 2017; Xu, Li, Chen, & Hou, 2015). From literature, it was found that most of research work are focusing on single objective optimisation while studies on multiobjective optimisation for such a robot design problem have been rarely presented.

Over the last decade, numerous MOMHs have been proposed (Gao, Zhou, Li, Pan, & Yi, 2015; Hidalgo-Paniagua, Vega-Rodríguez, & Ferruz, 2016; Liu, Zhang, He, & Jiang, 2018; Nuaekaew, Artrit, Pholdee, & Bureerat, 2017; Onan, Korukoğlu, & Bulut, 2016; Pholdee & Bureerat, 2013a,b; Qingfu & Hui, 2007; Robič & Filipič, 2005;

* Corresponding author.
E-mail addresses: sujbur@kku.ac.th (S. Bureerat), nantiwat@kku.ac.th (N. Pholdee), papoja@kku.ac.th (P. Jaroenapibal).

https://doi.org/10.1016/j.eswa.2019.06.033
0957-4174/© 2019 Elsevier Ltd. All rights reserved.

Russo, Bernardino, & Barbosa, 2018; Wang, Jiao, & Yao, 2015; Wang, Purshouse, & Fleming, 2013; Zareizadeh, Helfroush, Rahideh, & Kazemi, 2018; Zhang, Tian, & Jin, 2015). Some of effective and efficient MOMHs are an improved two-archive algorithm (Two_Arch2) (Wang et al., 2015), a preference-inspired co-evolutionary algorithm using goal vectors (PICEA-g), a knee point driven evolutionary algorithm for many-objective optimization (KnEA) (Zhang et al., 2015), an unrestricted population size evolutionary multiobjective optimisation algorithm (UPS-EMOA) (Aittokoski & Miettinen, 2010), hybridisation of real-code population-based incremental learning and differential evolution (MRPBIL-DE) (Pholdee & Bureerat, 2013b). Nevertheless, they have never been implemented with the robot trajectory planning problem. Moreover, most of research work which applied MOMHs for such a problem usually apply an individual algorithm to solve the problem while the comparative performance of many MHs has yet to be investigated. The MOMHs suitable for the robot trajectory planning multiobjective problem are still in question whereas inventing a new efficient MOMH for such a problem is an interesting topic.

This work presents the performance enhancement of a hybrid real code population based incremental learning and differential evolutionary algorithm by inventing a self-adaptive strategy for automatically tuning important optimisation parameters of the optimiser. The robot optimum trajectory planning has two objective functions as minimising trajectory time and jerk subject to velocity, acceleration, and jerk constraints of all joints. The design variables include joint positions and velocities at intermediate positions, moving time from initial to intermediate positions and that from intermediate to final positions. The performance of the proposed algorithm is validated against several efficient MOMHs including multi-objective evolutionary algorithm based on decomposition (MOEA/D) (Qingfu & Hui, 2007), Two_Arch2, PICEA-g, KnEA, UPS-EMOA, MRPBIL-DE, Differential Evolution for Multiobjective Optimization (DEMO) (Robič & Filipič, 2005), and NSGA-II. The rest of this paper include the sections of; hybrid real code population based incremental learning and differential evolutionary algorithm with adaptive parameter, trajectory planning optimisation problem, numerical experiment, results and discussions, and conclusions.

2. Hybrid real code population based incremental learning and differential evolutionary algorithm with adaptive parameter

Real code population based incremental learning (RPBIL) was proposed by Bureerat in 2011 (Bureerat, 2011) while it is extended to be a multiobjective version called, multiobjective real code population based incremental learning (MRPBIL) by Pholdee and Bureerat in 2013 (Pholdee & Bureerat, 2013b). The algorithm is based on a Pareto dominance approach (Coello Coello & Lechuga, 2002). The main idea of the algorithm is to use a real number population (a set of design solutions) rather than using a binary population as with binary-code multiobjective population-based incremental learning (MPBIL) (Bureerat & Sriworamas, 2007). Similarly to the binary code version, the MRPBIL used a number of probability matrices to represent a real-code population. Each probability matrix is called one tray while a set of probability trays ($\mathtt{P}$) contain a three-dimensional matrix sized $n \times n_I \times N_T$ where n, n_I and N_T are the number of design variables, the number of design domain subintervals and the number of trays, respectively. Each tray will be used to generate a subpopulation of design solutions which contains approximately N_P/N_T solutions where N_P is a population size.

Tables 1 and 2 show how the real population ($\mathtt{X}$) is generated by one probability tray of $\mathtt{P}$. The rows of the matrix corresponds to the elements of design variables (There are two design variable in the example) while the columns are associated with subintervals of the bounds of design variables (For this example, the design variable bounds are divided into 4 subintervals). The element P_{ij} represents the probability of having x_i in the subinterval j. A real-code population can be generated in a such way that $round(N_P.P_{ij})$ elements of x_i are generated at random inside the corresponding subinterval. Their column positions are then randomly permutated leading to a real-code subpopulation. In this example, the number of element x_1 to be randomly generated in the first subinterval is 1, and so on. Table 2 shows the resulting set of real-code design variables or a subpopulation created based on P_{ij} in Table 1. The subpopulation matrix, $\mathtt{X}$, containing real-code design vectors can eventually be written as:

$$\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_8\} = \begin{Bmatrix} 0.60 \\ 0.05 \end{Bmatrix}, \begin{Bmatrix} 0.25 \\ -0.80 \end{Bmatrix}, \begin{Bmatrix} 0.80 \\ -0.60 \end{Bmatrix}, \begin{Bmatrix} -0.15 \\ 0.40 \end{Bmatrix}, \begin{Bmatrix} 0.10 \\ 0.55 \end{Bmatrix}, \begin{Bmatrix} -0.75 \\ 0.35 \end{Bmatrix}, \begin{Bmatrix} 0.45 \\ -0.30 \end{Bmatrix}, \begin{Bmatrix} 0.30 \\ 0.90 \end{Bmatrix}$$

The search procedure of multiobjective RPBIL starts with initialised probability trays. A real-code population according to the trays are then created (Tables 1 and 2) while a Pareto archive is generated from sorting the members in the initial population. During optimisation search, if the number of non-dominated solutions exceeds the predefined archive size, the clustering technique is activated to remove some non-dominated solutions from the Pareto archive. Then, the probability trays are iteratively updated based on the non-dominated solutions. The archive is also replaced by the non-dominated solutions of the new population and the members in the previous archive. The process is repeated until a termination criterion is satisfied. Algorithm 1 shows computational steps for MRPBIL.

To update the probability tray, firstly, non-dominated solutions are divided into N_T groups by applying a clustering technique on the objective function domain. Then, the centroid of design variables of each group ($\mathbf{r}_G$) is evaluated and used to update the prob-

Table 1
A sample probability matrix.

Subintervals	$-1 \le x_i < -0.5$	$-0.5 \le x_i < 0$	$0 \le x_i < 0.5$	$0.5 \le x_i \le 1$
x_1	1/8	2/8	3/8	2/8
x_2	2/8	1/8	3/8	2/8

Table 2
Generate a real-code population randomly.

Subinterval	$-1 \le x_i < -0.5$	$-0.5 \le x_i < 0$	$0 \le x_i < 0.5$	$0.5 \le x_i \le 1$
x_1	−0.75	−0.25, −0.15	0.10, 0.30, 0.45	0.60, 0.80
x_2	−0.80, −0.60	−0.30	0.05, 0.40, 0.35	0.55, 0.90
Randomly permutate the positions in each row				
x_1	0.60, 0.25, 0.80, −0.15, 0.10, −0.75, 0.45, 0.30			
x_2	0.05, −0.80, −0.60, 0.40, 0.55, 0.35, −0.30, 0.90			

Algorithm 1
MRPBIL.

```
Input: number of generation ($N_G$), $n_I$, $N_T$, $N_p$ objective function name ($fun$), Pareto archive size ($N_A$)
Output: Pareto archive $\mathbf{A}$
Initialisation: $P_{ij} = n_I/N_p$ for each tray, a Pareto archive $\mathbf{A} = \{\}$
1: For $i = 1$ to $N_G$
2:     Generate a real code population $\mathbf{X}$ from the probability trays and evaluated objective function values, $\mathbf{f} = fun(\mathbf{X})$
3:     Select non-dominated solutions from the set $\mathbf{X} \cup \mathbf{A}$ and replace the old members of $\mathbf{A}$.
4:     Divide the non-dominated solutions into $N_T$ groups using a clustering technique, and find the centroid $\mathbf{r}_G$ of each group.
5:     Update a probability tray using Eqs. (1)-(4).
6:     If the number of members in $\mathbf{A}$ is larger than $N_A$, remove some of them using a clustering technique.
7: End
```

ability trays where matching $\mathbf{r}_G$ and a tray is carried out randomly. Eqs. (1)–(4) are used for updating an element in a probability tray;

$$P'_{ij} = (1 - L_R(j))P_{ij}^{old} + L_R(j) \tag{1}$$

$$L_R(j) = L_{R0} \exp\left(-(j - r)^2\right) \tag{2}$$

where L_{R0} and L_r are an initial learning rate and learning rate, respectively, and r is the subinterval number that the i-th element of $\mathbf{r}_G$ is located. Then, the updated probability matrix P'_{ij} is modified as

$$P''_{ij} = P'_{ij} / \sum_{j=1}^{n_I} P'_{ij} \tag{3}$$

in order to satisfy the condition

$$\sum_{j=1}^{n_I} P''_{ij} = 1 \tag{4}$$

where P''_{ij} is the final update of P_{ij}.

For the hybrid MRPBIL with DE (MRPBIL-DE), the algorithm is introduced by Pholdee and Bureerat in 2013 (Pholdee & Bureerat, 2013b). The hybridisation is achieved in such a way that, for each generation, a new population created by using the probability trays will be recombined with members in the Pareto archive by means of DE mutation and crossover (Storn & Price, 1997) before performing function evaluations. This concept is used in order to improve exploration capability in the optimiser. The search procedure is given in Algorithm 2 where F is a scaling factor, p_c is a DE crossover probability, and CR is probability of choosing elements of an offspring.

One of the weakness of MRPBIL-DE is that there are several optimisation parameters to be predefined before running. These parameters are crucial for its search performance. The best values of the parameters should result in the best performance of the algorithm which is usually problem-dependent. This is also the case for other MHs. As a result, a self-adaptive strategy for automatically tuning such parameters is used to cope with this problem. For the proposed self-adaptive MRPBIL-DE (abbreviated as A-MRPBIL-DE), the DE/best/1/bin and DE/best/2/bin reproduction strategies are used where the probability of each strategy being operated is decided by a parameter P_{Mu}. In selecting one pair or two pairs of solutions for mutation, not only the solutions in $\mathbf{X}$ can be selected but also those in the external archive $\mathbf{A}$ where the probability of choosing solutions from $\mathbf{A}$ is defined by P_{xr}. In addition, all parameter, $P_{M\text{u}}$, P_{xr}, F and CR are set to be self-adaptive.

The parameter $P_{M\text{u}}$, P_{xr}, F and CR are generated for each reproduction based on the information from the previous iterations. For each calculation, the parameter $P_{M\text{u}}$, P_{xr}, and CR are generated by mean of normal distribution randomisation (Tanabe & Fukunaga, 2013; Tanabe & Fukunaga, 2014; Zhang & Sanderson, 2009) based on their mean values of P_{Mum}, P_{xrm} and CR_{m} and their standard deviation values of 0.1 for all while the parameter F is generated by using a Cauchy distribution randomisation (Tanabe & Fukunaga, 2013; Tanabe & Fukunaga, 2014; J. Zhang & Sanderson, 2009) based on the mean values, F_{m} and the standard deviation values of 0.1. The parameters P_{Mum}, P_{xrm} F_{m} and CR_{m} are iteratively adapted using the following equations:

Algorithm 2
MRPBIL-DE.

```
Input: $N_G$, $n_I$, $N_P$, $N_T$, objective function name ($fun$), Pareto archive size ($N_A$)
Output: Pareto archive $\mathbf{A}$
Initialization: $P_{ij} = n_I/N_p$ for each tray, a Pareto archive $\mathbf{A} = \{\}$
            : Generate a real code population $\mathbf{X}$ from the probability trays and evaluate objective function values, $\mathbf{f} = fun(\mathbf{X})$
            : Find non-dominated solutions from $\mathbf{X} \cup \mathbf{A}$ and replace the old members in $\mathbf{A}$
1: For $i = 1$ to $N_G$
2:     Divide the non-dominated solutions into $N_T$ groups using a clustering technique, and find the centroid $\mathbf{r}_G$ of each group.
3:     Update a probability tray using $\mathbf{r}_G$.
4:     Generate a real code population $\mathbf{X}$ from the probability trays.
5:     For $j = 1$ to $N_P$ [recombine $\mathbf{X}$ and $\mathbf{A}$ using DE operators]
           5.1: Randomly select a solution $\mathbf{p}$ from the Pareto archive $\mathbf{A}$.
           5.2: Randomly select two solutions, $\mathbf{x}_{r1}$ and $\mathbf{x}_{r2}$ from the current population $\mathbf{X}$.
           5.3: Generate a new solution $\mathbf{c} = \mathbf{p} + F(\mathbf{x}_{r1} - \mathbf{x}_{r2})$
           5.4: Set $c_i$ into the interval $L_i \le c_i \le U_i$.
           5.5: If $rand < p_c$, perform crossover
               5.5.1: For $k = 1$ to $n$
               5.5.2: If $rand < CR$, $y_k = c_k$
               5.5.3: Otherwise, $y_{j,k} = p_k$
               5.5.4: End
           5.6: Save a newly generated solution in $\mathbf{Y}$
6:     End
7:     Evaluate objective function values of the new real code population $\mathbf{f} = fun(\mathbf{Y})$
8:     Find non-dominated solutions from $\mathbf{Y} \cup \mathbf{A}$ and update $\mathbf{A}$
9:     If the size of $\mathbf{A}$ is larger than $N_A$, remove some of the members in $\mathbf{A}$ using a clustering technique.
10: End
```

Algorithm 3
Self-adaptive MRPBIL-DE.

```
Input: $N_G$, $N_P$, $n_I$, $N_T$, objective function name (fun), Pareto archive size ($N_A$),
Output: $\mathbf{x}^{best}$, $f^{best}$
Initialization: $P_{ij} = n_I/N_p$ for each tray, a Pareto archive $\mathbf{A} = \{\}$
            : Generate a real code population $\mathbf{X}$ from the probability trays and evaluated objective function values, $\mathbf{f} = fun(\mathbf{X})$
            : Select non-dominated solutions from $\mathbf{X} \cup \mathbf{A}$ and save them to $\mathbf{A}$
            : $P_{\mathrm{Mum}}$=0.5, $P_{\mathrm{xrm}} = 0.5$, $F_{\mathrm{m}} = 0.5$, $CR_{\mathrm{m}} = 0.5$
            : $good_{PMu\mathrm{m}} = \{\}$, $good_{P\mathrm{xrm}} = \{\}$, $good_{\mathrm{CR}} = \{\}$ and $good_{\mathrm{F}} = \{\}$
1: For $i = 1$ to $N_G$
2:    Divide the non-dominated solutions into $N_T$ groups using a clustering technique, and find the centroid $\mathbf{r}_G$ of each group.
3:    Update the probability trays using $\mathbf{r}_G$.
4:    Generate a real code population $\mathbf{X}$ from the probability trays.
5:    For $j = 1$ to $N_P$ [recombine $\mathbf{X}$ and $\mathbf{A}$ using DE operators]
        5.1: Generate $P_{M\mathrm{u}}$, $P_{\mathrm{xr}}$, and $CR$ by normal distribution random and generate $F$ by
    Cauchy distribution random technique.
        5.2: Randomly select a solution $\mathbf{p}$ from Pareto archive $\mathbf{A}$.
        5.3: Randomly select four solutions, $\mathbf{x}_{\mathrm{r1}}$, $\mathbf{x}_{\mathrm{r2}}$,$\mathbf{x}_{\mathrm{r3}}$ and $\mathbf{x}_{\mathrm{r4}}$
            5.3.1 If $rand < P_{\mathrm{xr}}$, select the $\mathbf{x}_{\mathrm{r1}}$, $\mathbf{x}_{\mathrm{r2}}$,$\mathbf{x}_{\mathrm{r3}}$ and $\mathbf{x}_{\mathrm{r4}}$ from the current population $\mathbf{X}$.
            5.3.2 Otherwise, select the $\mathbf{x}_{\mathrm{r1}}$, $\mathbf{x}_{\mathrm{r2}}$,$\mathbf{x}_{\mathrm{r3}}$ and $\mathbf{x}_{\mathrm{r4}}$ from Pareto archive $\mathbf{A}$.
        5.4: Generate a new solution by using one of the mutation schemes.
            5.4.1 If $rand < P_{\mathrm{Mu}}$,$\mathbf{c} = \mathbf{p} + F(\mathbf{x}_{\mathrm{r1}} - \mathbf{x}_{\mathrm{r2}})$
            5.4.2 Otherwise, $\mathbf{c} = \mathbf{p} + F(\mathbf{x}_{\mathrm{r1}} + \mathbf{x}_{\mathrm{r2}} - \mathbf{x}_{\mathrm{r3}} + \mathbf{x}_{\mathrm{r4}})$
        5.5: Set $c_i$ into its bounds $L_i \le c_i \le U_i$.
        5.6: Perform crossover
            5.6.1: For $k = 1$ to $n$
            5.6.2: If $rand < CR$, $y_k = c_k$
            5.6.3: Otherwise, $y_{j,k} = p_k$
            5.6.4: End
        5.6: Save newly generated solution in $\mathbf{Y}$
6:    End
7:    Evaluate objective functions values of the new real code population $\mathbf{f} = fun(\mathbf{Y})$
8:    Select non-dominated solutions from $\mathbf{Y} \cup \mathbf{A}$ and save as $\mathbf{A}$. For each $\mathbf{y}_i \in \mathbf{Y}$that dominates at least one solution in $\mathbf{A}$, the parameters,
$P_{M\mathrm{u}}$, $P_{\mathrm{xr}}$, $CR$ and $F$ index $i$ are saved to $good_{PMu}$, $good_{P\mathrm{xr}}$, $good_{\mathrm{CR}}$ and $good_{\mathrm{F}}$, respectively.
9:    Update the parameters, $P_{\mathrm{Mum}}$, $P_{\mathrm{xrm}}$ $F_{\mathrm{m}}$ and $CR_{\mathrm{m}}$ using Eqs. (5)–(8)
10:    If the number of $\mathbf{A}$ larger than $N_A$, remove some of the members in $\mathbf{A}$ using a clustering technique.
11: End
```

$$P_{Mum}(T+1) = 0.9P_{Mum}(T) + 0.1mean(good_{PMu}), \tag{5}$$

$$P_{xrm}(T+1) = 0.9P_{xrm}(T) + 0.1mean(good_{Pxr}), \tag{6}$$

$$CR_m(T+1) = 0.9CR_m(T) + 0.1mean(good_{CR}), \tag{7}$$

$$F_m(T+1) = 0.9F_m(T) + 0.1\frac{sum\left(good_F^2\right)}{sum(good_F)} \tag{8}$$

where T is an iteration number. $good_{PMu}$, $good_{P\mathrm{xr}}$, $good_{\mathrm{CR}}$ and $good_{\mathrm{F}}$ are the archives for collecting the values of $P_{M\mathrm{u}}$, P_{xr}, F and CR that have been successfully used for generating a new solution. Successfully generated solutions are ones who dominate a solution in the Pareto archive ($\mathbf{A}$). The function $mean(good_X)$ means the average value of the members in the archive $good_X$ whereas $sum(good_F)$ and $sum(good^2{}_F)$ respectively represent the summation and sum square of all the elements in $good_F$. In this work, the weighting factors of 0.9 and 0.1 in Equations (5)–(8) are selected based on the successful record as reported in (Tanabe & Fukunaga, 2013, 2014; Zhang & Sanderson, 2009)

The search procedure for self-adaptive MRPBIL-DE is shown in Algorithm 3. The process starts with initialised probability trays, an empty Pareto archive, initialised values of P_{Mum}, P_{xrm} F_{m} and CR_{m}, and empty archives $good_{PMu}$, $good_{P\mathrm{xr}}$, $good_{\mathrm{CR}}$ and $good_{\mathrm{F}}$. The parameters $P_{M\mathrm{u}}$, P_{xr}, F and CR are then generated by normal distribution random and Cauchy distribution random technique and the real-code population is generated by the combination of RPBIL and DE operators in the same manner as the reproduction of MRPBIL-DE in Algorithm 2. After performing objective function evaluations, non-dominated solutions are then sorted from the combination of the newly generated solutions and the current Pareto archive. For a particular newly generated solution that dominates one of the members in the Pareto archive, the parameters $P_{M\mathrm{u}}$, P_{xr}, F and CR used to create this solution are saved to the $good_{PMu\mathrm{m}}$, $good_{P\mathrm{xrm}}$, $good_{\mathrm{CR}}$ and $good_{\mathrm{F}}$ archives. Then, the probability trays, the Pareto archive and the parameters P_{Mum}, P_{xrm} F_{m} and CR_{m} are iteratively updated until a termination criterion is satisfied.

3. Trajectory planning optimisation problem

Robot trajectory planning is the process to find a geometric trajectory of robot motion from starting position to final position subject to kinetic and dynamic constraints. The basic concept of finding a robot moving trajectory is to solve a non-linear motion equation based on position, velocity and acceleration constraints while some of conventional functions used for the non-linear motion equation are third order polynomial, five order polynomial, cubic spline, etc. For smooth movement of the robot, the trajectory can be divided into several connected small segments. The more segments would result in the smoother motions and consequently have more design variables for the motion equations. In this work, the technique used to find the motion part is adopted from (Savsani et al., 2013; Wichapong et al., 2018). To find the motion part of the robot, the complete trajectory is divided into two parts (or sub-paths) as from initial to intermediate positions and from the intermediate to final positions where the target points of the five order polynomial function is used to represent the motion equation for both parts (Savsani et al., 2013; Wichapong et al., 2018). It should be noted that the problem is set in order to get a smooth trajectory with minimum number of segments, therefore, only 2 segments are used. Moreover, the five order polynomial is selected for the segment's motion equation to obtain continuous acceleration.

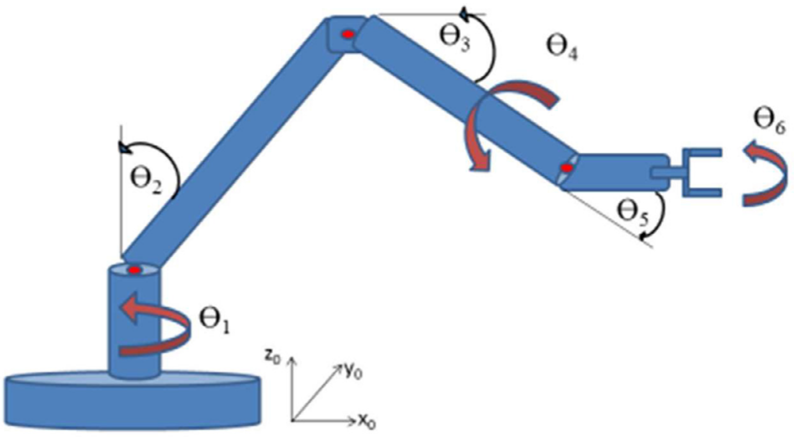


Fig. 1. This figure shows schematic diagram of the 6D robot.

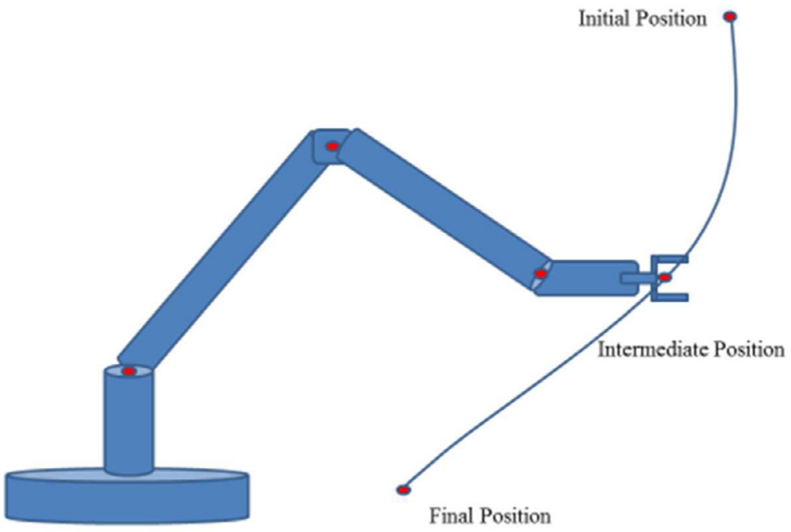


Fig. 2. This figure shows 3 positions on the trajectory.

The 6D robot as shown in Fig. 1 is employed as the case study for this work while Fig. 2 shows an example of motion path which is divided into two parts. To find the motion equations, the initial joint angle and final joint angle have their predefined coordinate pose (θ_i, θ_f) while the velocity and acceleration of initial and final positions are set to be zero. For continually motion, the two sub-paths will have the same position, velocity and acceleration at the intermediate points.

For the initial to the intermediate position sub-path, the motion equation is given by Eq. (9)

$$\theta_{i,i+1}(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \tag{9}$$

The parameters $a_{0,\dots}$, a_5 are polynomial constant coefficients which can be calculated based on the motion constraints by using Eq. (10),

$$\begin{aligned}
\theta_i &= a_0,\ \theta_{i+1} = a_0 + a_1 t_1 + a_2 {t_1}^2 + a_3 {t_1}^3 + a_4 {t_1}^4 + a_5 t_1^5 \\
\dot{\theta}_i &= a_1,\ \dot{\theta}_{i+1} = a_1 + 2a_2 t_1 + 3a_3 {t_1}^2 + 4a_4 {t_1}^3 + 5a_5 t_1^4 \\
\ddot{\theta}_i &= 2a_2,\ \ddot{\theta}_{i+1} = 2a_2 + 6a_3 t_1 + 12a_4 {t_1}^2 + 20a_5 t_1^3
\end{aligned} \tag{10}$$

where t_1 is the time which the arm moves from the initial position (i) to the intermediate position ($i+1$).

For the intermediate to final position sub-path, the motion equation is given by Eq. (11)

$$\theta_{i+1,f}(t) = b_0 + b_1 t + b_2 t^2 + b_3 t^3 + b_4 t^4 + b_5 t^5 \tag{11}$$

The parameters $b_{0,\dots}$, b_5 are polynomial constant coefficients which can be calculated based on the motion constraints by using Eq. (12),

$$\begin{aligned}
&\theta_{i+1} = b_0 + b_1 t_1 + b_2 {t_1}^2 + b_3 {t_1}^3 + b_4 {t_1}^4 + b_5 t_1^5, \\
&\theta_f = b_0 + b_1 t_2 + b_2 {t_2}^2 + b_3 {t_2}^3 + b_4 {t_2}^4 + b_5 t_2^5, \\
&\dot{\theta}_{i+1} = b_1,\ \dot{\theta}_f = b_1 + 2b_2 t_2 + 3b_3 {t_2}^2 + 4b_4 {t_2}^3 + 5b_5 t_2^4, \\
&\ddot{\theta}_{i+1} = 2b_2 + 6b_3 t_1 + 12b_4 {t_1}^2 + 20b_5 t_1^3 \\
&\ddot{\theta}_f = 2b_2 + 6b_3 t_2 + 12b_4 {t_2}^2 + 20b_5 t_2^3
\end{aligned} \tag{12}$$

where t_2 is the time for the robot arm moving from the intermediate position (i+1) to the final position (f).

The multiobjective optimisation problem is posed to minimise the travelling time and jerk subjected to motion (velocity, acceleration and jerk) constraints. The problem can be formulated as;

$$Min:\ f_1 = \text{Travelling times} = t_1 + t_2,$$

$$f_2 = \text{Maximum jerk} = \max\left(\left(\dddot{\theta}_{j=1}, \dots, \dddot{\theta}_{j=6}\right)\right)$$

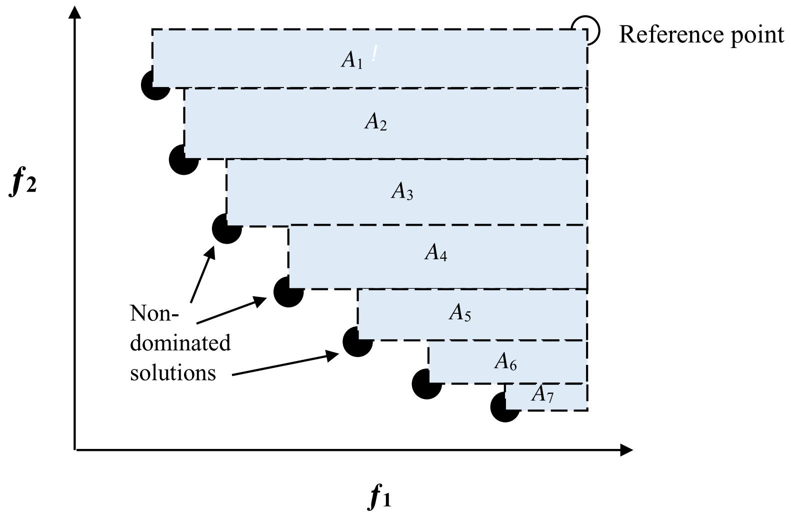


Fig. 3. Hypervolume of a bi-objective problem.

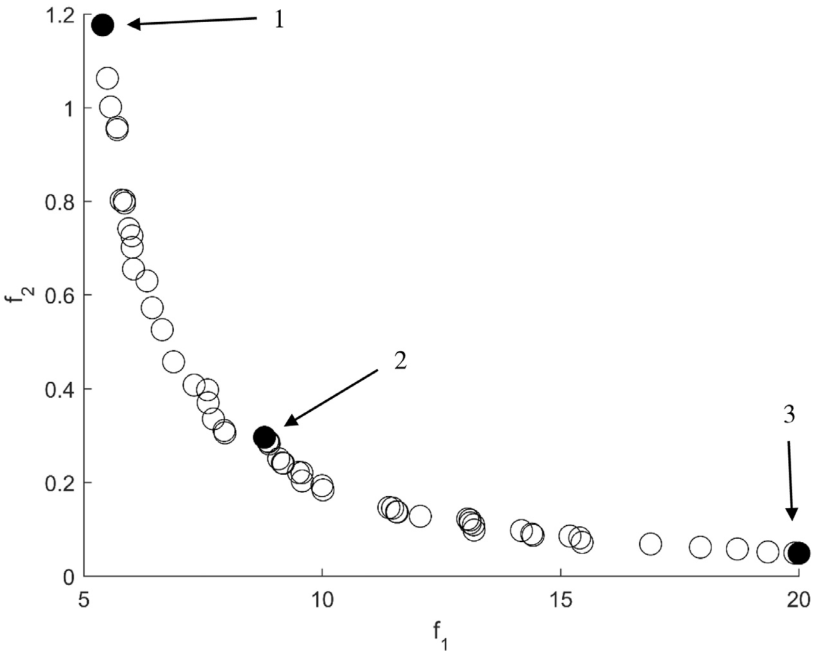


Fig. 4. Pareto plot of the AMRPBIL-DE.

Table 3
Joint angular positions in each joint of the robot manipulator.

Joint	$\theta_i(deg)$	$\theta_f(deg)$
1	−10	55
2	20	35
3	15	30
4	150	10
5	30	70
6	120	25

Table 4
Constraints for each joint.

Joint	$\omega_c(deg/s)$	$\alpha_c(deg/s^2)$	$J_c(deg/s^3)$
1	100	60	60
2	95	60	66
3	100	75	85
4	150	70	70
5	130	90	75
6	110	80	70

Subject to

$$\left|\dot{\theta}_j\right| \le \omega_c$$

$$\left|\ddot{\theta}_j\right| \le \alpha_c$$

$$\left|\dddot{\theta}_j\right| \le J_c$$

where ω_c = angular velocity constraint for joint j

α_c = acceleration constraint for joint j
J_c = angular jerk constraint for joint j

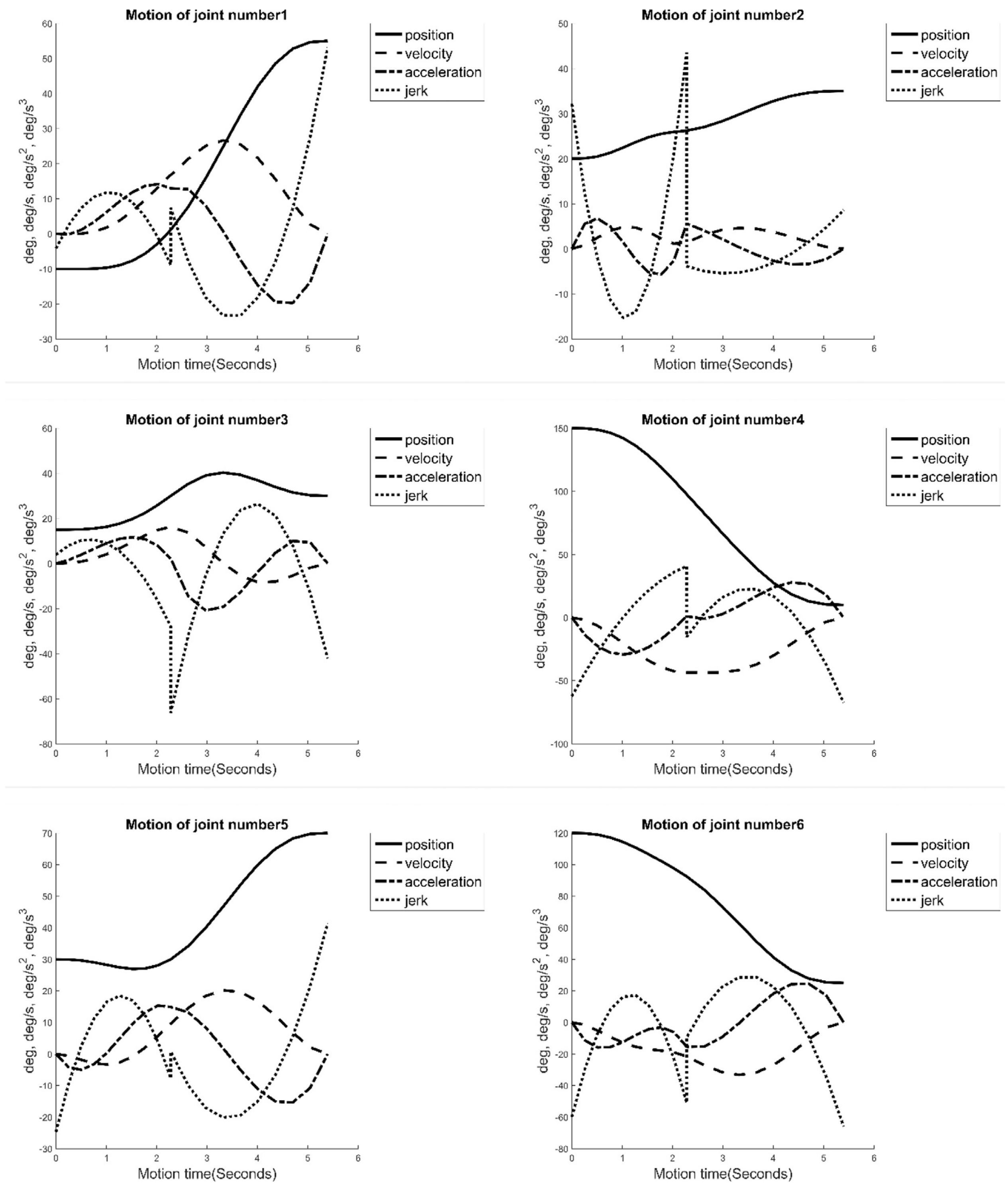


Fig. 5. Motion plot of the point number 1 in Fig. 4.

The predefined values for the initial and final points for a test problem for finding trajectory and constraint values for each joint are detailed in Tables 3 and 4 (Tangpattanakul & Artrit, 2009).

The design variables are the traveling time from the initial to intermediate positions, that from the intermediate to final positions, and intermediate position, velocity, and acceleration, which can be expressed as:

$$\mathbf{x} = \left\{t_1, t_2, \theta_{i+1,j=1}, \ldots, \theta_{i+1,j=n},\ \dot{\theta}_{i+1,j=1}, \ldots, \dot{\theta}_{i+1,j=n}, \ldots, \ddot{\theta}_{i+1,j=1}, \ldots, \ddot{\theta}_{i+1,j=n}\right\}$$

where n is the number of robot joints which is set to be 6. The total number of design variables is 20.

4. Numerical experiment

To investigate the performance of the proposed self-adaptive MRPBIL-DE, eight MOMHs along with the proposed algorithm are used to solve the optimisation problem detailed in the previous section. Several self-adaptive MRPBIL-DE based on different

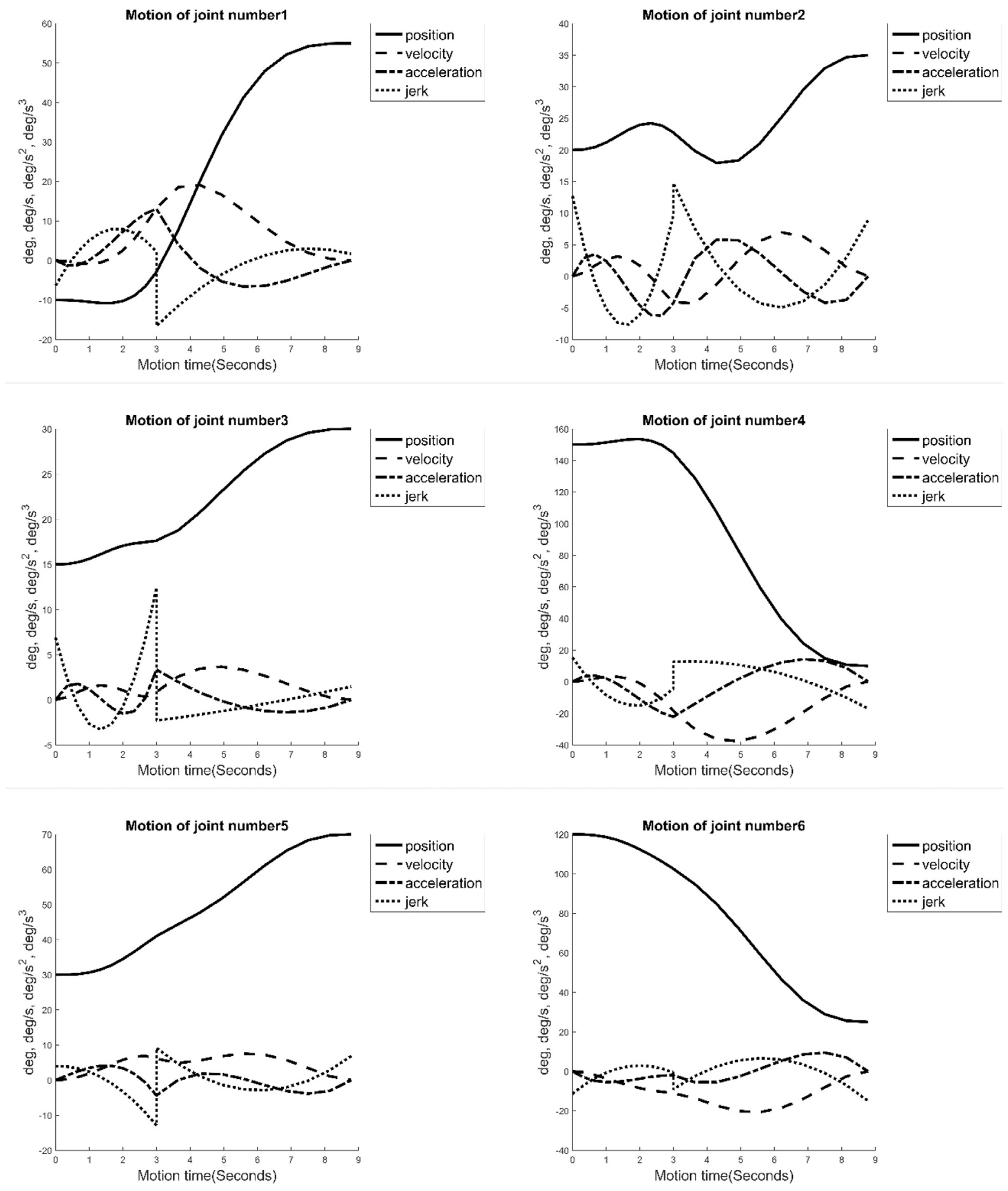


Fig. 6. Motion plot of the point number 2 in Fig. 4.

number of subinterval is used in comparison. Those MOMHs and their optimisation parameter settings (details of notations can be found in the corresponding reference of each method) are detailed in Table 5 (Pholdee & Bureerat, 2013b; Pholdee, Bureerat, Jaroenapibal, & Radpukdee, 2017):

Each algorithm is employed to tackle the optimisation problem for 30 independent runs. The population size is set to be $N_P = 100$ with the total number of generations being 200, thus, the number of function evaluations is 100×200. The fuzzy set penalty function is applied to cope with design constraints (Pholdee & Bureerat, 2014). A hypervolume indicator is used to assess the search performance of the MOMHs.

The definition of the hypervolume indicator is an amount of volume (for 3D) or area (for 2D) covered by non-dominated solutions measured with respect to a predefined reference point. For 2D problems, the hypervolune values is the area as shown in

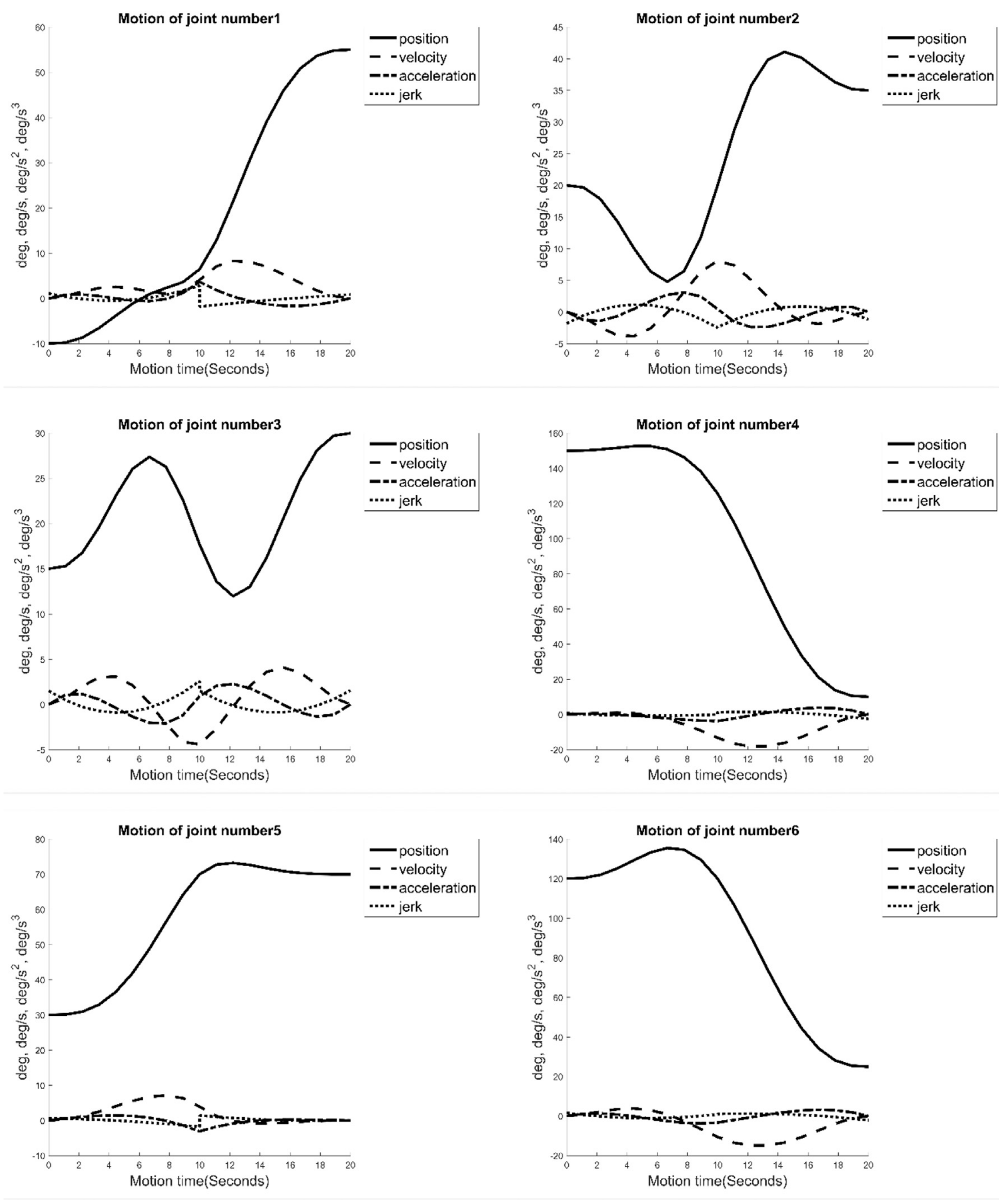


Fig. 7. Motion plot of the point number 3 in Fig. 4.

Fig. 3 which can be calculated by

$$H_v = \sum_{i=1}^{N_A} A_i$$

where H_{v-} is the hypervolume value and A_i is the area as shown in the figure (Pholdee & Bureerat, 2013b).

5. Results and discussions

Having carried out 30 optimization runs of 8 MOMHs along with the proposed A-MRPBIL-DE on solving the 6-D robot trajectory multiobjective optimisation problem, the performance is investigated based on the hypervolume indicator (Pholdee & Bureerat, 2013b). Table 6 shown the normalized hypervolume values obtained from all optimisers. From the table, the mean values of the hypervolume values (Mean) is used to measure the algorithm search convergence and consistency. In cases that two methods have similar hypervolume mean values, the standard deviation of the hypervolume values (STD) is used to decide the more consistent method. The higher Mean the better convergence rate and search consistency. From Table 6, the results show that all A-MRPBIL-DE versions are superior to the other MOMHs. The best convergence rate is from using A-MRPBIL-DE5 while the second

Table 5
MOMHs used in this study and their parameter setting.

Optimisers	Parameter setting
MOEA/D	Number of neighbouring weight vectors = 6, crossover probabilities = 1.0, mutation probabilities = 0.1
Two-Arch2	Use the code from Wang et al. (2015), crossover probability = 1 (default values), mutation probability = 0.1 (default values).
PICEA-g	Employ the code from Wang et al. (2013), all optimization parameters are set as default values from Wang et al. (2013)
KnEA	Use the code from Zhang et al. (2015), all optimization parameters are set as default values from Zhang et al. (2015).
UPS-EMOA	crossover probability = 0.7 scaling factor = [0.6, 0.9] probability of choosing element from offspring in crossover = 0.5 minimum population size = 10 burst size = 25
MRPBIL-DE	initial learning rate = 0.25, mutation probability = 0.05, mutation shift = 0.20, crossover probability = 0.7, scaling factor for differential evolution (DE) operator (F) = 0.8, probability of choosing elements from offspring in crossover (CR) = 0.5.
DEMO	scaling factor for differential evolution (DE) operator (F) = 0.5, crossover probability in DE algorithm (CR) = 0.2.
NSGA-II	crossing-over probability = 0.7, mutation probability = 0.4
A-MRPBIL-DE1:	Algorithm 3, $N_T = \lceil N_p/10 \rceil$
A-MRPBIL-DE2:	Algorithm 3, $N_T = \lceil N_p/7 \rceil$
A-MRPBIL-DE3:	Algorithm 3, $N_T = \lceil N_p/5 \rceil$
A-MRPBIL-DE4:	Algorithm 3, $N_T = \lceil N_p/4 \rceil$
A-MRPBIL-DE5:	Algorithm 3, $N_T = \lceil N_p/3 \rceil$
A-MRPBIL-DE6:	Algorithm 3, $N_T = \lceil N_p/2 \rceil$

Table 6
Normalized hypervolume values obtained from all optimisers.

Optimises	Mean	STD	Max.	Min.
MOEA/D	0.0843	0.1173	0.3501	0.0000
Two-Arch2	0.7491	0.0229	0.7792	0.6866
MRPBIL-DE	0.7064	0.0373	0.7540	0.5908
PICEA-g	0.7417	0.0148	0.7651	0.7080
KnEA	0.7535	0.0103	0.7685	0.7179
DEMO	0.7527	0.0059	0.7628	0.7390
UPS-EMOA	0.7564	0.0494	0.7882	0.5155
NSGA-II	0.7483	0.0143	0.7695	0.7157
A-MRPBIL-DE1	0.7630	0.0101	0.7802	0.7409
A-MRPBIL-DE2	0.7657	0.0100	0.7795	0.7446
A-MRPBIL-DE3	0.7619	0.0132	0.7801	0.7312
A-MRPBIL-DE4	0.7646	0.0114	0.7799	0.7366
A-MRPBIL-DE5	0.7675	0.0093	0.7859	0.7475
A-MRPBIL-DE6	0.7618	0.0116	0.7763	0.7252

best and the third best are A-MRPBIL-DE2 and A-MRPBIL-DE4, respectively. For the measure of search consistency based on STD, the most consistent is A-MRPBIL-DE5 while the second most consistent is A-MRPBIL-DE2. UPS-EMOA obtained the highest maximum hypervolume values while A-MRPBIL-DE5 obtained the highest minimum hypervolume values.

Table 7 shows the comparison of all optimisers based on the statistical *t*-test (Pholdee & Bureerat, 2016). Since there are 14 optimisers implemented, a comparison matrix sized 14×14 whose elements are full of zeros is first generated. The null hypothesis for the *t*-test is that, for a particular test problem, the mean of hypervolume values obtained from running method *I* is not different from that obtained from using method *J* at the 5% significance level. If the null hypothesis is rejected and the mean from method *J* is higher, the element *IJ* of the matrix is set to be 1. After summing up all values in the matrix columns, the best optimiser is the one that has the highest score. The summation and the ranking are also given in the table. It was found that the A-MRPBIL-DE5 is the best method while the second best is A-MRPBIL-DE1, A-MRPBIL-DE2, A-MRPBIL-DE3, A-MRPBIL-DE4, and A-MRPBIL-DE6, which are the in the same rank.

Based on this study, with the assigned total number of function evaluations, the algorithm with self-adaptive optimisation parameters perform better than the algorithms that require predefined optimisation parameters (MRPBIL-DE versus A-MRPBIL-DE). Based on the use of different numbers of subintervals (A-MRPBIL-DE1, . . ., A-MRPBIL-DE6), it was found that the results obtained from using various numbers of subintervals are not significantly different except for the number of subinterval of $N_T = \lceil N_p/3 \rceil$ (A-MRPBIL-DE5) which is the best performer in this study. Overall, the proposed A-MRPBIL-DE is said to be the best performer for solving the 6D robot trajectory plaining multiobjective optimisation.

Fig. 4 shows the plot of the Pareto front of the best run obtained from A-MRPBIL-DE5 while Figs. 5–7 shows plot of 3 selected solutions from the front. From the figures, it was illustrated that if the moving time increases, the jerk obtained will decrease and vice versa. The solution having shorter moving time (Points 1) leads to higher jerks while the solution points which having longer moving time (Points 3) give the lower jerks. From the motion plots of solution point 3 which has long moving time, the jerk obtained is close to zero while the motion plot of solution point 1 that has shorter moving time shows higher jerk. For the solution 2, it is the trade-off between solutions 1 and 3. From the results obtained, it is shown that A-MRPIBL-DE5 successfully solve the multiobjective optimisation of the robot trajectory problem. Within a single run, several solutions which are the trade-off of the objective functions are obtained providing a set of optimal solutions to be selected best suitable for design requirements.

Table 7
Comparison of hypervolumn values based on statistical *t*-test.

Optimises	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)
MOEA/D (1)	0	1	1	1	1	1	1	1	1	1	1	1	1	1
Two-Arch2 (2)	0	0	0	0	0	0	0	0	1	1	1	1	1	1
MRPBIL-DE (3)	0	1	0	1	1	1	1	1	1	1	1	1	1	1
PICEA-g (4)	0	0	0	0	1	1	0	0	1	1	1	1	1	1
KnEA (5)	0	0	0	0	0	0	0	0	1	1	1	1	1	1
DEMO (6)	0	0	0	0	0	0	0	0	1	1	1	1	1	1
UPS-EMOA (7)	0	0	0	0	0	0	0	0	0	0	0	0	0	0
NSGA-II (8)	0	0	0	0	0	0	0	0	1	1	1	1	1	1
A-MRPBIL-DE1 (9)	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A-MRPBIL-DE2 (10)	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A-MRPBIL-DE3 (11)	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A-MRPBIL-DE4 (12)	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A-MRPBIL-DE5 (13)	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A-MRPBIL-DE6 (14)	0	0	0	0	0	0	0	0	0	0	0	0	1	0
Sum	0	2	1	2	3	3	2	2	7	7	7	7	8	7
Rank	14	9	13	9	7	7	9	9	2	2	2	2	1	2

6. Conclusions

In this work, self-adaptive MRPBIL-DE is successfully present for a 6D multiobjective robot trajectory planning. Several well-establish MOMHs are used to compare for identifying the performance of the proposed algorithm. After performing several optimisation runs on solving the problem, it is shown that the proposed self-adaptive MRPBIL-DE perform better than all other MOMHs used in this study while the self-adaptive MRPBIL-DE which having the number of subintervals of $N_T = \lceil N_p/3 \rceil$ (A-MRPBIL-DE5) is the best performer. The Pareto front obtained from A-MRPBIL-DE5 shows several good trade-off solutions between the two objective functions. The results obtained from this study are set as the baseline for further studies of robot trajectory planning multiobjective optimisation. Also, performance the robot trajectory planning optimisation which has more than 2 object functions should be further investigated.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Credit authorship contribution statement

Sujin Bureerat: Methodology, Software, Resources, Supervision, Writing - review & editing. **Nantiwat Pholdee:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing - original draft. **Thana Radpukdee:** Data curation, Visualization, Writing - original draft. **Papot Jaroenapibal:** Visualization, Writing - review & editing.

Acknowledgements

The authors are thoroughly grateful for the financial support provided by the KKU Engineering Research Fund, the Faculty of Engineering, Khon Kaen University, and Thailand Research Fund (TRF), grant number MRG6180278.

References

Aittokoski, T., & Miettinen, K. (2010). Efficient evolutionary approach to approximate the Pareto-optimal set in multiobjective optimization, UPS-EMOA. *Optimization Method and Software, 25*(6), 841–858.
Bureerat, S. (2011). Improved population-based incremental learning in continuous spaces. In A. Gaspar-Cunha, R. Takahashi, G. Schaefer, & L. Costa (Eds.). In *Soft computing in industrial applications: 96* (pp. 77–86). Berlin Heidelberg: Springer.
Bureerat, S., & Sriworamas, K. (2007). Population-based incremental learning for multiobjective optimisation. *Advance in Soft Computing, 9*, 223–231.
Coello Coello, C. A., & Lechuga, M. S. (2002). MOPSO: A proposal for multiple objective particle swarm optimization. In *Proceedings of the 2002 Congress on evolutionary computation: vol. 2* (pp. 1051–1056). CEC'02 (Cat. No.02TH8600). doi:10.1109/CEC.2002.1004388.
Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197. doi:10.1109/4235.996017.
Gao, L., Zhou, Y., Li, X., Pan, Q., & Yi, W. (2015). Multi-objective optimization based reverse strategy with differential evolution algorithm for constrained optimization problems. *Expert Systems with Applications*, *42*(14), 5976–5987. https://doi.org/10.1016/j.eswa.2015.03.016.
Garg, D. P., & Kumar, M. (2002). Optimization techniques applied to multiple manipulators for path planning and torque minimization. *Engineering Applications of Artificial Intelligence, 15*(3-4), 241–252.
Hidalgo-Paniagua, A., Vega-Rodríguez, M. A., & Ferruz, J. (2016). Applying the MOVNS (multi-objective variable neighborhood search) algorithm to solve the path planning problem in mobile robotics. *Expert Systems with Applications*, *58*, 20–35. https://doi.org/10.1016/j.eswa.2016.03.035.
Huang, J., Hu, P., Wu, K., & Zeng, M. (2018). Optimal time-jerk trajectory planning for industrial robots. *Mechanism and Machine Theory*, *121*, 530–544. https://doi.org/10.1016/j.mechmachtheory.2017.11.006.
Wichapong, K., Bureerat, S., & Pholdee, N. (2018). Trajectory planning of a 6D robot based on Meta Heuristic algorithms. In *MATEC web of conference: 220* (p. 06004).
Liu, J., Zhang, H., He, K., & Jiang, S. (2018). Multi-objective particle swarm optimization algorithm based on objective space division for the unequal-area facility layout problem. *Expert Systems with Applications*, *102*, 179–192. https://doi.org/10.1016/j.eswa.2018.02.035.
Mac, T. T., Copot, C., Tran, D. T., & De Keyser, R. (2016). Heuristic approaches in robot path planning: A survey. *Robotics and Autonomous Systems, 86*, 13–28.
Mac, T. T., Copot, C., Tran, D. T., & Keyser, R. D. (2017). A hierarchical global path planning approach for mobile robots based on multi-objective particle swarm optimization. *Applied Soft Computing*, *59*, 68–76. https://doi.org/10.1016/j.asoc.2017.05.012.
Nuaekaew, K., Artrit, P., Pholdee, N., & Bureerat, S. (2017). Optimal reactive power dispatch problem using a two-archive multi-objective grey wolf optimizer. *Expert Systems with Applications*, *87*(Supplement C), 79–89. https://doi.org/10.1016/j.eswa.2017.06.009.
Onan, A., Korukoğlu, S., & Bulut, H. (2016). A multiobjective weighted voting ensemble classifier based on differential evolution algorithm for text sentiment classification. *Expert Systems with Applications*, *62*, 1–16. https://doi.org/10.1016/j.eswa.2016.06.005.
Pholdee, N., & Bureerat, S. (2013a). Hybrid real-code population-based incremental learning and approximate gradients for multi-objective truss design. *Engineering Optimization*, 1–20. doi:10.1080/0305215x.2013.823194.
Pholdee, N., & Bureerat, S. (2013b). Hybridisation of real-code population-based incremental learning and differential evolution for multiobjective design of trusses. *Information Sciences*, *223*, 136–152. doi:10.1016/j.ins.2012.10.008.
Pholdee, N., & Bureerat, S. (2014). Comparative performance of meta-heuristic algorithms for mass minimisation of trusses with dynamic constraints. *Advances in Engineering Software, 75*, 1–13.
Pholdee, N., & Bureerat, S. (2016). Hybrid real-code ant colony optimisation for constrained mechanical design. *International Journal of Systems Science, 47*(2), 474–491.
Pholdee, N., Bureerat, S., Jaroenapibal, P., & Radpukdee, T. (2017). Many-objective optimisation of trusses through meta-heuristics. *Lecture Notes in Computer Science, 10261*, 143–152.

Piazzi, A., & Visioli, A. (2000). Global minimum-jerk trajectory planning of robot manipulators. IEEE Transactions on Industrial Electronics, *47*(1), 140–149.

Pires, E. J. S., de Moura Oliveira, P. B., & Machado, J. A. T. (2007). Manipulator trajectory planning using a MOEA. *Applied Soft Computing, 7*(3), 659–667. https://doi.org/10.1016/j.asoc.2005.06.009.

Qingfu, Z., & Hui, L. (2007). MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation, 11*(6), 712–731. doi:10.1109/tevc.2007.892759.

Robič, T., & Filipič, B. (2005). DEMO: differential evolution for multiobjective optimization. In C. Coello Coello, A. Hernández Aguirre, & E. Zitzler (Eds.). In *Evolutionary multi-criterion optimization: 3410* (pp. 520–533). Springer Berlin Heidelberg.

Russo, I. L. S., Bernardino, H. S., & Barbosa, H. J. C. (2018). Knowledge discovery in multiobjective optimization problems in engineering via Genetic Programming. *Expert Systems with Applications, 99*, 93–102. https://doi.org/10.1016/j.eswa.2017.12.008.

Savsani, P., Jhala, R., & Savsani, V. J. (2013). Optimized trajectory planning of a robotic arm using teaching learning based optimization (TLBO) and artificial bee colony (ABC) optimization techniques. *Paper presented at the Systems Conference (SysCon)*. 2013 IEEE International.

Storn, R., & Price, K. (1997). Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization, 11*(4), 341–359. doi:10.1023/a:1008202821328.

Tanabe, R., & Fukunaga, A. (2013). Evaluating the performance of SHADE on CEC 2013 benchmark problems. *the Evolutionary Computation (CEC), 20-23 June 2013 2013 IEEE Congress on*.

Tanabe, R., & Fukunaga, A. S. (2014). Improving the search performance of SHADE using linear population size reduction. *Paper presented at the IEEE Congress on Evolutionary Computation (CEC), 2014, 6-11 July 2014*.

Tangpattanakul, P., & Artrit, P. (2009). Minimum-time trajectory of robot manipulator using Harmony Search algorithm. *the Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2009. ECTI-CON 2009. 6th International Conference on*.

Wang, H., Jiao, L., & Yao, X. (2015). Two_Arch2: An improved two-archive algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation, 19*(4), 524–541.

Wang, R., Purshouse, R. C., & Fleming, P. J. (2013). Preference-inspired coevolutionary algorithms for many-objective optimization. *IEEE Transactions on Evolutionary Computation, 17*(4), 474–494. doi:10.1109/tevc.2012.2204264.

Xu, Z., Li, S., Chen, Q., & Hou, B. (2015). MOPSO based multi-objective trajectory planning for robot manipulators. *Paper presented at the 2015 2nd International Conference on Information Science and Control Engineering, 24-26 April 2015*.

Zareizadeh, Z., Helfroush, M. S., Rahideh, A., & Kazemi, K. (2018). A robust gene clustering algorithm based on clonal selection in multiobjective optimization framework. *Expert Systems with Applications, 113*, 301–314. https://doi.org/10.1016/j.eswa.2018.06.047.

Zhang, J., & Sanderson, A. C. (2009). JADE: Adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation, 13*(5), 945–958. doi:10.1109/tevc.2009.2014613.

Zhang, X., Tian, Y., & Jin, Y. (2015). A knee point-driven evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation, 19*(6), 761–776. doi:10.1109/tevc.2014.2378512.