Applied Soft Computing 70 (2018) 680–700

Contents lists available at ScienceDirect

Applied Soft Computing

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

A multi-objective particle swarm optimization algorithm based on dynamic boundary search for constrained optimization

Mohamad Zihin bin Mohd Zain [a], Jeevan Kanesan [a,*], Joon Huang Chuah [a], Saroja Dhanapal [b], Graham Kendall [c]

[a] University of Malaya, Department of Electrical Engineering, Faculty of Engineering, 50603, Kuala Lumpur, Malaysia
[b] University of Malaya, Expert System and Optimization Group, Faculty of Engineering, 50603, Kuala Lumpur, Malaysia
[c] University of Nottingham, School of Computer Science, Jubilee Campus, Nottingham NG8 1BB, UK

ARTICLE INFO

Article history:
Received 23 December 2017
Received in revised form 27 May 2018
Accepted 11 June 2018
Available online 20 June 2018

Keywords:
Multi-objective particle swarm optimization
Swarm intelligence
Constrained multi-objective optimization
Fed-batch fermentation
Tumor treatment

ABSTRACT

Due to increased search complexity in multi-objective optimization, premature convergence becomes a problem. Complex engineering problems poses high number of variables with many constraints. Hence, more difficult benchmark problems must be utilized to validate new algorithms performance. A well-known optimizer, Multi-Objective Particle Swarm Optimizer (MOPSO), has a few weakness that needs to be addressed, specifically its convergence in high dimensional problems and its constraints handling capability. For these reasons, we propose a modified MOPSO (M-MOPSO) to improve upon these aspects. M-MOPSO is compared with four other algorithms namely, MOPSO, Multi-Objective Grey Wolf Optimizer (MOGWO), Multi-Objective Evolutionary Algorithm based on Decompositions (MOEA/D) and Multi-Objective Differential Evolution (MODE). M-MOPSO emerged as the best algorithm in eight out of the ten constrained benchmark problems. It also shows promising results in bioprocess application problems and tumor treatment problems. In overall, M-MOPSO was able to solve multi-objective problems with good convergence and is suitable to be used in real world problem.

1. Introduction

One of the attributes of most real-world engineering is that they often have multiple conflicting goals. These multiple objectives often provide certain trade-offs which result in many solutions which might be acceptable to the end user. In a single-objective optimization problem, the optimal solution is apparently defined. Oppositely, in a multi-objective problem, there is no direct way to define the superiority of one solution compared to another. One of the ways to address this is by applying Pareto dominance and Pareto optimality concepts where there exists more than one 'optimal solution'.

In the past two decades, several single-objective swarm intelligent (SI) based optimization methodologies such as Particle Swarm Optimization (PSO) [1], Evolutionary Algorithm (EA) [2], and Grey Wolf Optimizer (GWO) [3] have been utilized to address multi-objective problems. The conversion to Multi-Objective Particle Swarm Optimization (MOPSO) [4], Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) [5] and Multi-Objective Grey Wolf Optimizer (MOGWO) [6] were carried out by introducing new concepts such as decomposition and Pareto dominance.

PSO, in particular, has been intensively studied in recent years as many multi-objective versions of PSO have been introduced. Coello and Pulido [4] approached MOPSO by incorporating the concept of Pareto dominance to address several multi-objective optimization test functions and comparing the results with a few evolutionary algorithms. Tripathi and Bandyopadhyay [6] proposed an adaptive approach by implementing time variant inertia and acceleration coefficients in their algorithm called TV-MOPSO. Goh et al. [7] adapted a competitive and cooperative co-evolutionary technique to MOPSO in their algorithm called Competitive and Cooperative Co-Evolutionary Multi-Objective Particle Swarm Optimization Algorithm (CCPSO). Xue and Zhang [8] studied the application of MOPSO in feature selection problem and proposed two versions of MOPSO called NSPSOFS and CMDPSOFS by implementing the con-

* Corresponding author.
E-mail addresses: mzihin_91@siswa.um.edu.my (M.Z.b. Mohd Zain), jievan@um.edu.my (J. Kanesan), jhchuah@um.edu.my (J.H. Chuah), saroja.dhanapal@um.edu.my (S. Dhanapal), graham.kendall@nottingham.ac.uk (G. Kendall).

https://doi.org/10.1016/j.asoc.2018.06.022
1568-4946/

cept of non-dominated sorting for the former and the concept of crowding, mutation, and dominance for the latter. A new hybrid optimizer was proposed by Cheng and Zhan [9] by integrating an innovative local optimal particles search strategy into MOPSO. Meza and Espitia [10] presented Multi-Objective Vortex Particle Swarm Optimization (MOVPSO) which is based on rotational and translational motions of a swarm.

All the proposed algorithms performed well in solving the problems presented in their respective papers. However, due to increased search complexity in multi-objective optimization, premature convergence becomes a problem [11]. This complexity increases with an increasing number of variables, which are exhibited by many engineering problems. Hence, more complex benchmark problems with higher dimensionality need to be utilized to validate the performance of multi-objective optimization algorithms.

For these reasons, we propose the modified MOPSO (M-MOPSO), which is loosely based on MOPSO proposed by Coello and Pulido [4], by retaining some elements used in the original MOPSO but introducing new processes to either replace or complement the original procedures. Our new methodology aims to improve the original MOPSO in solving constrained problems, enhancing exploitation and changing the way exploration takes place, to help escape from local optima through the introduction of dynamic search boundary. This is carried out by replacing the traditional swarm intelligence approach which is based on the trajectory of swarms following a leader based and by varying velocity of each individual. We also propose an improved archiving procedure to that which is currently used in MOPSO. The algorithm is verified by addressing the CEC 2009 benchmark problems, which are some of the most challenging benchmark problems. We also address application problems drawn from the domain of fed-batch fermentation and tumor treatment.

In fed-batch fermentation, nutrient feeding along the bioprocess increases product concentrations. Controlled nutrient supply increases biomass in a controlled manner and this improves product concentrations with less impact of product and/or nutrient inhibition of the biomass. This complex nature of fed-batch fermentation encourages optimization methodology development that predicts an optimal feeding profile to enhance process performance. In order to simulate the processes, differential equations which represent the mass balances of different state variables are developed. The fed-batch fermentation model may have a single or multiple objectives. Apart from the usual maximization of product yield, a multi-objective version of fed-batch fermentation models can also consist of the optimization of various objectives such as substrate utilization, environmental impact and economic benefits. Some works that have utilized multi-objective swarm intelligence in multi-objective fed-batch fermentation or bioprocess problems have been published e.g. [12–15]. The success of these metaheuristics in tackling these types of problems inspired us to apply the proposed M-MOPSO to bioprocess problems.

In this paper, we also address tumor treatment planning by determining the good quality chemotherapy treatment solutions. This problem deals with the trade-off between the reduction of tumor cells and the amount of medicine used. This trade-off requires Pareto solutions to determine the suitable treatment profile. Though many researchers addressed this problem using optimal control theory, most of them used weighting scheme-based optimal control theory as they consider the problem as a single objective problem [16–19]. Due to the multi-objective attributes of the chemotherapy treatment problem solved in this work, the various multi-objective swarm intelligence techniques such as Multi-Objective Optimization Differential Evolution (MODE), MOPSO and MOGWO are used to compare with M-MOPSO to address the tumor treatment planning.

The rest of the paper is organized as follows. Section 2 presents a literature review by briefly describing the concept of multi-objective optimization, PSO and MOPSO. Section 3 describes the proposed M-MOPSO algorithm. Section 4 explains the methodology used in our experiments. Section 5 presents the results of the experiments along with a discussion. Section 6 concludes our work and suggests some future directions for the research.

2. Multi-objective optimization

A multi-objective optimization problem with a number of competing objectives can be defined as follows:

$$Minimize: F(X) = f_1(X), f_2(X), \ldots, f_G(X), \tag{1}$$

$$Subject\ to: R_i^{lower} \le x_i \le R_i^{upper},\ i = 1, 2, \ldots, d \tag{2}$$

where G is the number of objective, d is the number of variables and $\left[R_i^{lower}, R_i^{upper}\right]$ are the boundaries of i th variables. In Pareto dominance, given that there are two candidate solutions: $Y = (y_1, y_2, \ldots, y_d)$ and $Z = (z_1, z_2, \ldots, z_d)$, vector Y dominates vector Z (denoted as $Y \succ Z$) if and only if, the objective function value of vector Y is less than or equal to the objective function value of vector Z in all the G objective space, and the objective function value of vector Y is less than to the objective function value of vector Z in at least one of the G objective space, as formulated in Eqs. (3) and (4).

$$f_g(Y) \le f_g(Z),\ \forall g \in \{1, \ldots, G\} \tag{3}$$

$$f_g(Y) < f_g(Z),\ \forall g \exists \{1, \ldots, G\} \tag{4}$$

Solution Y is considered as a non-dominated Pareto optimal solution if it is not dominated by any other solutions. No better solutions than Y exist in the particular problem. However, similarly good solutions may exist, dependent on user perception. A solution Y, which is an element of X, ($Y \in X$) is called Pareto-optimal if and only if, there does not exist a solution Z, which is an element of X, that dominates any solution Y, as formulated in Eq. (5).

$$\nexists Z \in X | f(Z) \succ f(Y) \tag{5}$$

The Pareto optimal set is defined by a set of solutions that fulfil Eqs. (3) and (4), while at the same Eq. (5) holds true. The collective fitness values obtained from these solutions are known as the Pareto front or trade-off surface.

2.1. Particle swarm optimization (PSO)

In 1995, Kennedy and Eberhart developed Particle swarm optimization (PSO) [1], an algorithm inspired from swarm behaviour such as exhibited in fish and bird schooling. This observation motivated a recent research direction called swarm intelligence. In PSO, the solutions are known as particles and are generated using real-number randomness. These particles behave as a swarm through global communication. By adjusting the flight direction of each particle in a quasi-stochastic manner, the swarm is able to search the space of an objective function for near optimal solutions. The flight directions or trajectories are actually the piecewise paths created by positional vectors of the particles. The pattern of movement for each particle is achieved by incorporating global and local search in a hybrid of stochastic and deterministic manner. The information of the global best position g^* is shared among all particles while at the same time, each particle also retains its own history of best position found as x_j^*. The value of g^* and x_j^* guide the search of each particle and point its destination. Simultaneously, element of randomness is also incorporated in the movements of particles towards their destination. During its search, a particle will update its value of personal best position whenever it found a better location than the

last best position. At any time t during iterations, all n particles have distinct individual current best position and unique current global best position. The particles keep on finding and improving the global best position until they saturate or the maximum number of iterations are achieved.

The essential steps of the particles in PSO can be summarized in the equation below:

$$v_j^{t+1} = v_j^t + \alpha R_1 \cdot \left[g^* - x_j^t\right] + \beta R_2 \cdot [x_j^* - x_j^t] \quad (6)$$

where x_j^t and v_j^t are the position and velocity vector for particle j at time t respectively. R_1 and R_2 are two random vectors with values between 0 and 1. The constants α and β are the two learning parameters in PSO. If $\alpha > \beta$, the particle has higher tendency to converge towards the global best position g^* instead of its best individual position x_j^* and vice versa. Each particle update its own velocity in each iteration using the equation above. The velocity is initialized as zero and the new position is updated as follows:

$$x_j^{t+1} = x_j^t + v_j^{t+1} \quad (7)$$

Various experimentations show that PSO outperforms other traditional algorithms for addressing optimization problems. The reason for this can be attributed to its ability to convey the information of current best estimates among the swarm which leads to an improved and faster convergence towards the optimality. One of its drawbacks is that it does not have memory. PSO does not store the historical trajectories of each particle in each iteration. This makes it hard for the algorithm to avoid local optima.

2.2. Multi-objective particle swarm optimization (MOPSO)

One of the earlier attempts to solve multi-objective problems using PSO was made by Coello and Pulido [4] using Multi-objective Particle Swarm Optimization (MOPSO). The algorithm uses the concept of Pareto dominance to find solutions for multi-objective problems. It also employs a secondary population or external archive to store non-dominated solutions and guides the search of future generations. A special mutation operator is also implemented to improve the search procedure.

2.3. MOPSO procedures

The MOPSO algorithm is as follows:

1 Initialize the population, X^j for $j = 1, 2, \ldots, n$; where n is the number of population.
2 Initialize the speed, VEL^j for each particle as follow:

$$VEL^j = 0$$

3 Evaluate each particle.
4 Store non-dominated solutions in archive/repository, *REP*.
5 Generate hypercubes.
6 Initialize the memory of each particle by storing initial X^j positions as best found positions so far, BFP^j as follow:

$$BFP^j = X^j$$

7 Compute the speed of each particle as follow:

$$VEL^j = W \times VEL^j + R_1 \times \left(BFP^j - X^j\right) + R_2 \times (REP - X^j)$$

where W (inertia weight) takes a value of 0.4; R_1 and R_2 are random numbers in the range [0. . .1]
8 Compute the new positions of each particle as follow:

$$X^j = X^j + VEL^j$$

9 Maintain the particles within the search boundaries.
10 Evaluate each particle.
11 Apply mutation to each particle.
12 Update *REP* and hypercubes by inserting non-dominated solutions into the repository and eliminate dominated solutions from the repository.
13 Update each particle memory by replacing the previous best position with the current best position found by each particle.
14 If maximum iteration is achieved, terminate. Otherwise repeat step 7.

MOPSO uses Pareto ranking scheme to handle multi-objective optimization problems. The algorithm store previously generated non-dominated solutions by recording the history of best solutions found by a particle.

There are two components in the external repository used by MOPSO which are the archive controller and the grid. The archive controller functions as the decision-maker for the addition and deletion of solutions in the archive. In every iteration, each non-dominated solutions found by the primary population are compared with every solution in the external repository. The grid system used in MOPSO is a modification of the adaptive grid. In this system, each time new solutions are inserted into the archive, the grid space will adapt to accommodate solutions that lie outside the boundary of current grid. The adaptive grid is an objective function space which is divided into regions. This space is formed by hypercubes and has as many components as objective functions. The mutation operator in MOPSO is designed in such a way to have high amplitude of search coverage across every design variables in earlier iteration while gradually shrinking the coverage over time by using a non-linear function.

3. Modified multi objective particle swarm optimization (M-MOPSO)

M-MOPSO shares similarities with MOPSO. In terms of the utilization of external archive/repository (*REP*), in the original MOPSO, the repository is made up of two main elements: the archive controller and the grid. The archive controller governs the selection and removal of the repository members. The grid system used in MOPSO is in the form of adaptive hypercubes where the objective space is divided into several regions to store the solutions. This system is used to reduce the computational cost when the archive controller needs to add or remove the repository member. Though the same principle is used in M-MOPSO, the execution is different. While the M-MOPSO uses the same grid system, the procedure for its archive controller is modified in several ways. These modifications, along with the introduction of other new procedures are described in the following section:

3.1. Boundary control mechanism

M-MOPSO is initialized by randomly generating n number of particles in the population X within the problem's upper and lower boundary. The value of n is predetermined by the user.

$$X = \begin{bmatrix} x_1^1 & \cdots & x_d^1 \\ \vdots & \ddots & \vdots \\ x_1^n & \cdots & x_d^n \end{bmatrix} \text{ or } \quad X^j = \begin{bmatrix} x_1^j, x_2^j, \ldots, x_d^j \end{bmatrix} \tag{8}$$

where x_i^j is the variable in i th dimension of j th individual. Initially, the variables of each particle are generated randomly below the middle range of its scope as shown in Eq. (9). This decision is made to balance between the convergence speed and the population diversity of the algorithm. Taking up the full range of each particle's scope from the beginning of the search will theoretically improve population diversity of an algorithm, but considering the dynamic boundary mechanism implemented in M-MOPSO, the population diversity is preserved not only during the beginning but throughout the whole run without sacrificing the exploitation aspect.

$$x_i^j = R_i^{lower} + rand \times Q_i^j,$$
$$where \quad rand \sim \cup([0,\ 1]), Q_i = \left| \frac{R_i^{upper} - R_i^{lower}}{2} \right|, \ i = 1,\ 2, \ldots,\ d, \tag{9}$$

where d is the number of variables, n is the pre-determined number of population, R_i^{upper} and R_i^{lower} are the upper and lower boundary respectively.

A new boundary control mechanism is introduced in this paper. This mechanism is the main evolutionary process of the proposed algorithm. In each iteration t, each individual in the population will produce a new individual according to the current boundary of each individual. The boundary of each individual changes dynamically by taking into account the current position of the individual and the boundary factor, b_f which is defined in Section 3.3. This new procedure is implemented to improve the exploitation aspect of M-MOPSO. With this new approach, balanced exploration and exploitation can be achieved by intelligently expanding or shrinking the search boundary of each individual based on a set of conditions. This boundary is determined by calculating the initial value of Q as follow:

$$Q_i = \frac{R_i^{lower} - R_i^{upper}}{2}, \quad i = 1,\ 2,\ \ldots,\ d \tag{10}$$

where R_i^{upper} and R_i^{lower} is the problem's upper and lower boundary respectively. The value of Q will shrink in each iteration as follows:

$$Q_i^{t+1} = Q_i^t \times s_{f2},\ i = 1,\ 2,\ \ldots,\ d \tag{11}$$

where s_{f2} is the predetermined parameter called shrink factor. The upper and lower individual boundary at iteration t is calculated as follow:

$$UB_i^j = x_i^j + Q_i^j,\ i = 1,\ 2,\ \ldots,\ d,\ j = 1,\ 2,\ \ldots,\ n \tag{12}$$

$$LB_i^j = x_i^j - Q_i^j,\ i = 1,\ 2,\ \ldots,\ d,\ j = 1,\ 2,\ \ldots,\ n \tag{13}$$

where x_i^j is the position in i th dimension of j th individual. The values that exceed the specified problem boundary will be replaced with their respective boundary value. Each individual will produce a new individual, x'^j_i as follows:

$$x'^j_i = LB_i^j + rand \times \left(UB_i^j - LB_i^j\right),\ i = 1,\ 2,\ \ldots,\ d,\ j = 1,\ 2,\ \ldots,\ n \tag{14}$$

where $rand \sim \cup([0,\ 1])$. Each new individual will be evaluated and its fitness is equal to its objective function value. For each individual, some of the variables (the position in each dimension) have the probability to become the value of its respective boundaries as follows:

$$\begin{aligned} &\text{IF } x'^j_i < 0.1 \times rand \times \left|R_i^{upper} - R_i^{lower}\right| + R_i^{lower} \\ &\qquad x'^j_i = R_i^{lower} \\ &\text{ELSE IF } x'^j_i > R_i^{upper} - 0.1 \times rand \times \left|R_i^{upper} - R_i^{lower}\right| \\ &\qquad x'^j_i = R_i^{upper} \\ &\text{END IF} \end{aligned} \tag{15}$$

where $rand \sim \cup([0,\ 1])$. This is to ensure that the variables that are close to the value of their boundaries have the probability to go to their boundaries, hence improving exploitation. The value of Q may change to simulate abrupt boundary expansion or shrinking following this condition:

$$\begin{aligned} &\text{IF } Q_i < b_f \times \left|R_i^{upper} - R_i^{lower}\right| + R_i^{lower} \\ &\qquad Q_i = OQ_i \times b_f \\ &END\ IF \end{aligned} \tag{16}$$

where OQ_i is the initial Q_i value obtained in Eq. (10). The determination of the boundary factor, b_f is explained in Section 3.2.

3.2. Boundary factor, b_f determination

M-MOPSO employs a new dynamic boundary mechanism to search for new solutions. Each individual will evolve based on its respective boundaries. These boundaries may shrink or expand in each iteration depending on the boundary factor, b_f. Smaller b_f ensures greater exploitation while larger b_f encourages greater exploration. The b_f is determined as follows:

1 Initially, b_f is calculated as follows:

$$b_f = rand \tag{17}$$

Where $rand \sim \cup([0,\ 1])$.

2 If the number of functional evaluations (NFE) is more than half the predetermined maximum NFE, b_f is calculated as follow:

$$b_f = 0.1 \times rand \tag{18}$$

where $rand \sim \cup([0,\ 1])$

3 If the number of REP member is equal to the predetermined maximum repository size, $nREP$ for $\frac{nREP}{2}$ number of iterations, b_f is determined using Eq. (17).

4 After S_{f1} number of iterations from the previous step, where S_{f1} is a predetermined parameter called saturation factor, b_f is determined using Eq. (18).

5 Repeat step 3.

3.3. Repository members admittance method

In the original MOPSO, the archive controller needs to determine the domination of each of repository member every time new members are admitted. This can lead to high computational cost especially when the size of the repository grows larger in each iteration. Besides, identical solutions or particles (X) in the population may be admitted into the finite-sized repository, causing it to rapidly saturate. In M-MOPSO, several changes are made to lower the computational cost and ensure the uniqueness of the solutions in the repository. The pseudocode for repository member admittance is as follows:

1. Determine each particle X domination.
2. Assign all REP member as non-dominated.
3. FOR each nondominated X
   FOR each REP member
   BREAK if current X is identical to current REP
   Assign current REP as dominated if it is dominated by current X
   ELSE assign current X as dominated if it is dominated by current REP and BREAK
   END FOR
   END FOR
4. Insert nondominated REP and nondominated X into REP.

3.4. Repository member deletion method

Each time new non-dominated solutions are found, the archive controller will add them into the repository. However, if the size of the repository exceeds the maximum allowable, some members of the repository will be removed by the archive controller. The original MOPSO determines which member to be removed based on the density of the grids. Members in the grid with higher density have a higher chance of being removed. In M-MOPSO, the factor that determines the removal of a member depends on the Euclidean distance in the objective space between each repository member

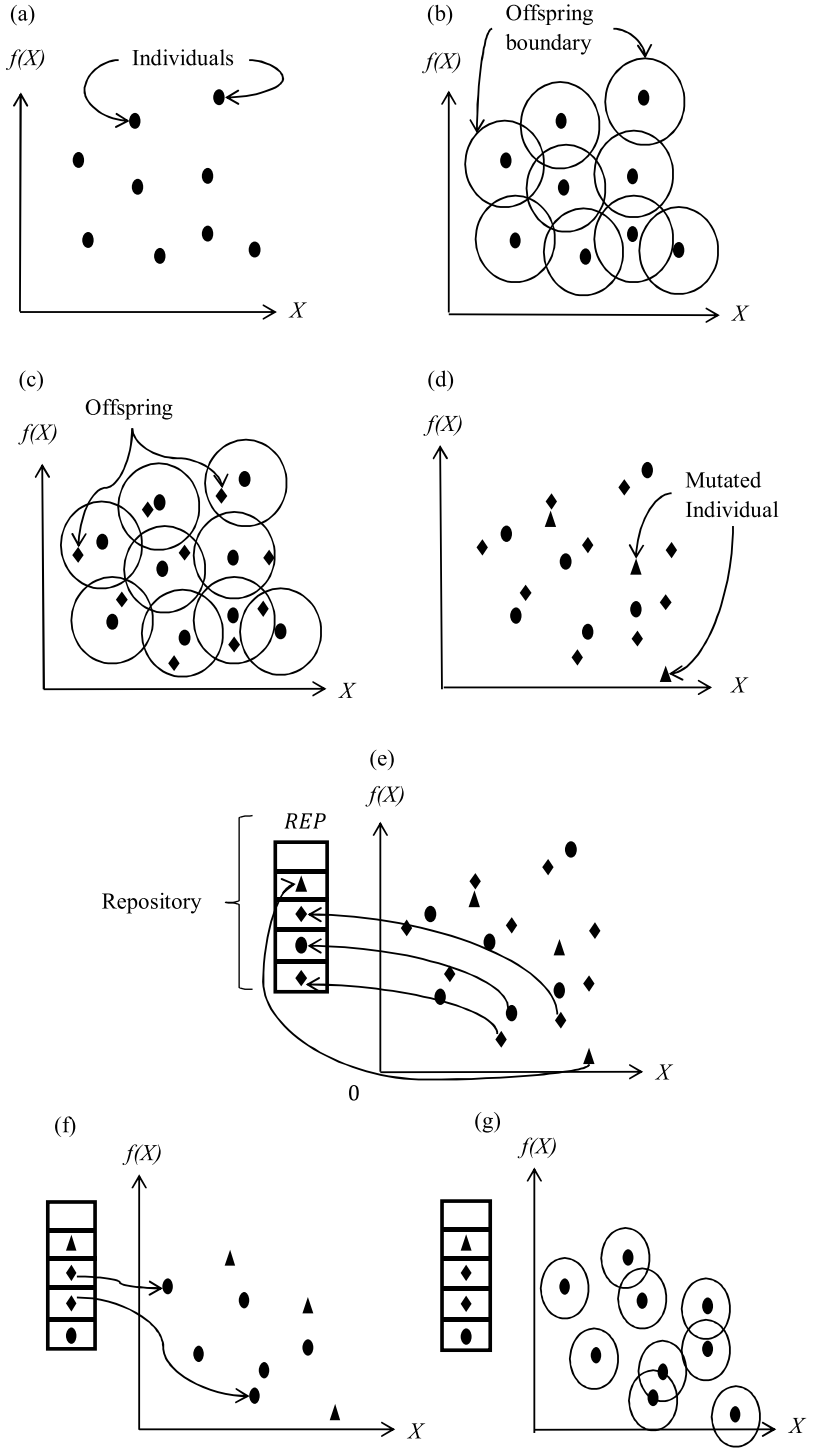


Fig. 1. (a) Initialization. (b) Generate offspring boundaries. (c) Generate offspring. (d) Mutate population (e) Update repository. (f) Update population. (g) Update boundary.

and the latest admitted member. The repository member deletion method in M-MOPSO is as follows:

1 We use roulette-wheel selection to select one member. The weight, W is defined as the Euclidean distance (in objective space) between each *REP* member and the latest admitted *REP* member (nearer member has a higher weight). The calculation of the roulette-wheel probabilities for member deletion, P is as follow:

$$P = e^{\gamma * W} \tag{19}$$

where e is an exponential function and γ is a pre-defined parameter called deletion selection pressure. Higher value of γ will results in higher member deletion probability. This formula is used by the original MOPSO [4] and MOGWO [6].

2 Delete the selected member.

3 Repeat step 1 until the number of *REP* members does not exceed the maximum allowable number.

3.5. Mutation operator and population update method

In M-MOPSO, the old population is replaced through the means of mutation. The same mutation operator used by MOPSO is also used in M-MOPSO. As discussed at the end of Section 2.3, the mutation operator tries to explore with all the particles at the beginning of the search before rapidly decreasing (with respect to the number of iterations) the number of particles that are affected by the mutation operator. The same applies to the range of the design variables of each particles that are affected by the mutation operator i.e., the number of design variables disturbed or mutated reduced over time. The pseudocode is as follows:

```
% particle = particle to be mutated
% dims = number of dimensions (i.e., decision variables)
% currentgen = current iteration
% totgen = total number of iterations
% mutrate = mutation rate
FUNCTION Mutation – Operator (particle, dims, currentgen, totgen, mutrate)
  BEGIN
    IF flip ((1 - currentgen / totgen)^(5/mutrate)) THEN
    BEGIN
      wichdim = random (0, dims - 1)
      mutrate = (upperbound[wichdim] - lowerbound[wichdim]) *
                (1 - currentgen/totgen)^(5/mutrate)
      ub = particle[wichdim] + mutrange
      lb = particle[wichdim] - mutrange
      IF lb < lowerbound[wichdim] THEN lb = lowerbound[wichdim]
      IF ub > lowerbound[wichdim] THEN ub = upperbound[wichdim]
      particle[wichdim] = RealRandom(lb, ub)
  END IF
END FUNCTION
```

The difference between the implementation of mutation in MOPSO and M-MOPSO is in the timing of its execution. In MOPSO, mutation occurs after the new population was generated, while in M-MOPSO, mutation occurs on the old population only. If the mutation process cannot improve the individual, its saturation counter is incremented by one. Once the saturation counter reaches a predetermined value, that individual may be replaced by one of the following method (the method used is chosen randomly):

- Choose one *REP* member by roulette-wheel selection. Members from less crowded areas in the grid have higher probability of being selected. The calculation of the probabilities, P is as follow:

$$P = e^{-\beta * W} \tag{20}$$

where β is a pre-defined parameter called leader selection pressure.

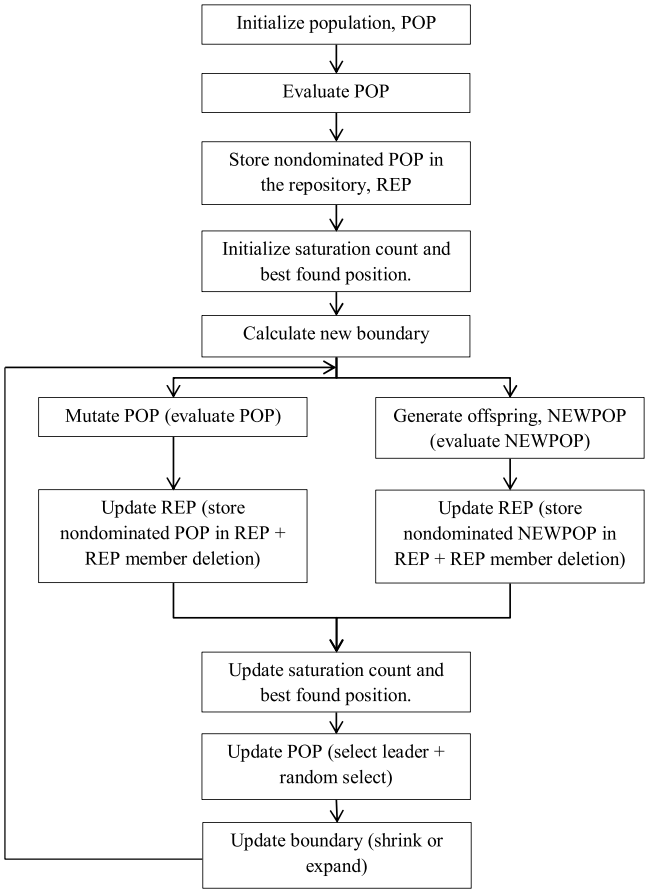


Fig. 2. M-MOPSO's flowchart.

- Randomly select one *REP* member.

Fig. 1(a–g) illustrate the mechanism of M-MOPSO. Fig. 1(a) shows the initialization of each individual in the population, represented by the dots. In Fig. 1(b), the boundaries shown as circular lines around each individual, within which new offspring will be produced, are generated by each individual. In Fig. 1(c), each individual generates an offspring, represented by the diamonds, within their respective boundary. Fig. 1(d) shows the mutated population indicated by the triangles. The mutated individuals will only replace their non-mutated counterpart if they dominate their non-mutated counterpart. Fig. 1(e) shows the repository update mechanism where the non-dominated individuals of the current population and the current non-dominated offspring are inserted in the repository. Fig. 1(f) shows the population update procedure where saturated individuals are replaced by a solution from the repository. In Fig. 1(g), each individual boundary in the population is updated.

3.6. M-MOPSO procedures

The flowchart for M-MOPSO is given in Fig. 2 and the algorithm is as follows:

1 Randomly generate initial population, X^j for $j = 1, 2, \ldots, n$ within the boundary.

$$x_i^j = R_i^{lower} + rand \times Q_i^j,$$

$$\text{where} \quad rand \sim \cup([0, 1]), Q_i = \left| \frac{R_i^{upper} - R_i^{lower}}{2} \right|, \; i = 1, 2, \ldots, d$$

where d is the number of variables, n is the pre-determined number of population, R_i^{upper} and R_i^{lower} are the upper and lower boundary respectively.

2 Evaluate $f(X^j)$, store non-dominated solutions in archive/repository, *REP*, generate hypercubes, initialize saturation count, S^j to zero, store initial X^j positions as best found positions so far, BFP^j.

3 Generate new boundary.

$$UB_i^j = x_i^j + Q_i^j$$

$$LB_i^j = x_i^j - Q_i^j$$

4 Each particle in the population generates offspring, X'^j within new boundary.

$$x'^j_i = LB_i^j + rand \times \left| UB_i^j - LB_i^j \right|$$

where $rand \sim \cup([0, 1])$

5 For each offspring, some of the variables have the probability to become the value of their respective boundaries as follows:

$$\begin{aligned}
&\text{IF } x'^j_i < 0.1 \times rand \times \left| R_i^{upper} - R_i^{lower} \right| + R_i^{lower} \\
&\qquad x'^j_i = R_i^{lower} \\
&\text{ELSE IF } x'^j_i > R_i^{upper} - 0.1 \times rand \times \left| R_i^{upper} - R_i^{lower} \right| \\
&\qquad x'^j_i = R_i^{upper} \\
&\text{END IF}
\end{aligned}$$

where $rand \sim \cup([0, 1])$

6 Evaluate $f(X'^j)$ and store nondominated X'^j in *REP*, update *REP* by removing dominated solutions, update hypercubes.

7 X^j performs mutation to generate X''^j. Evaluate $f(X''^j)$ and replace X^j if X''^j is better.

8 Compare X^j with BFP^j and update BFP^j. Increment S^j if better BFP^j is not found, otherwise reset S^j to zero.

9 Replace all X^j with respective BFP^j.

10 Store non-dominated X^j in *REP*, update *REP* by removing dominated solutions, update hypercubes.

11 If $S^j = S_{f1}$, replace X^j by population update method.

12 Shrink current boundary.

$$Q_i = Q_i \times S_{f2}$$

13 Determine boundary factor, b_f.

14 If $Q_i < b_f \times \left| R_i^{upper} - R_i^{lower} \right| + R_i^{lower}$, update the boundary.

$$Q_i = OQ_i \times b_f$$

where OQ_i is the original Q_i value obtained in Step 1.

15 If maximum iteration is achieved, terminate. Otherwise repeat step 3.

3.7. Similarities and differences between MOPSO and M-MOPSO

M-MOPSO has similarities with the original MOPSO. Both are population-based evolutionary algorithm, which evolve the solution vector/population (*POP*) in each iteration. Like MOPSO, M-MOPSO also uses an external population/repository (*REP*) to store the non-dominated *POP* and construct the Pareto front. M-MOPSO also employs identical adaptive grid mechanism used in MOPSO. In addition, both utilize elitism to update an individual's memory. Finally, M-MOPSO uses the same mutation operator utilized by MOPSO, though the mutation itself is different. The six differences between MOPSO and M-MOPSO are detailed in Table 1.

4. Methodologies

In this work, the proposed M-MOPSO was coded in MATLAB 8.1 (R2012a) and all simulations were run on Windows 7 platform using Intel i7-920, 2.67 GHz processor with 6 GB RAM.

4.1. Benchmark problems

We compare our algorithm with four other multi-objective algorithms namely multi-objective grey wolf optimizer (MOGWO), multi-objective particle swarm optimizer (MOPSO), and multi-objective evolutionary algorithm based on decompositions (MOEA/D) and multi-objective differential evolution (MODE). Ten constrained multi-objective test problems (CF1-CF10) proposed in CEC 2009 considered very difficult are used [20] to evaluate the algorithms. These problems provide convex, non-convex, discontinuous and multi-modal multi-objective search spaces with different Pareto optimal fronts. For all problems, 100 search agents are utilized and the algorithms are run for a maximum of 300,000 function evaluations. The number of parameters or variables for each of the constrained test functions is 10. For all algorithms, the constraints are handled by using a penalty method. The penalty is added to the objective function as follows:

$$fx = fx + \left| \min \begin{bmatrix} 0 & p \end{bmatrix} * 10 \right| \tag{21}$$

where p is the value of the constraint function.

The parameters for all algorithms are set at default values as recommended by their respective authors [4,5,21]. The two parameters used for M-MOPSO are as follows:

- $S_{f1} = 10$: saturation factor
- $S_{f2} = 0.97$: shrink factor

In M-MOPSO, these two parameters influence the convergent behaviour of the algorithm. These parameter values were fine-tuned and in our experiment, we found them after carrying out a set of initial experiments.

S_{f1} is a predetermined parameter called saturation factor. It determines when a particle or an individual in the population is considered saturated. This parameter is used twice in the algorithm: during the boundary factor determination and during updating the population. Larger value of S_{f1} will cause a particle to persist longer (takes up more iteration before being replaced). Larger value of S_{f1} will also encourage more exploration as detailed in Section 3.2.

S_{f2} is a predetermined parameter called shrink factor. It determines the general shrinking rate of each particle's search boundary in each dimension. The recommended range for this parameter is between 0 and 1, where higher value emphasize exploitation at the expanse of mitigated exploration and convergence speed, and vice-versa.

The parameters used for MOPSO are as follows:

Table 1
Differences between MOPSO & M-MOPSO.

Differences	Original MOPSO	Modified MOPSO (M-MOPSO)
1. Population evolution procedure	MOPSO utilizes swarm intelligence technique where the whole population coordinates their movements by following a single leader. This is achieved by considering the current leader position, the memory of each individual (individual best position) and their velocity.	M-MOPSO instead employs boundary control mechanism. This is achieved by deliberate shrinking and expansion of each individual boundary. New populations are randomly generated within these adaptive boundaries. M-MOPSO does not restrict its search direction by following a single leader but instead each individual evolves independently according to their own boundary.
2. Mutation procedure	In MOPSO, the mutation procedure is applied after the new position of the population has been found. The new population will replace the old one.	In M-MOPSO, the mutation is applied directly on the old population (before evolution). The new population never replaces the old one. The old population is only replaced by mutation or the population update procedure, where the repository member may replace the old population when it became saturated.
3. Repository update procedure	MOPSO updates the contents of its repository only after the new population has both been generated and mutated.	M-MOPSO updates the contents of its repository after the old population was mutated and after the new population has been generated.
4. Repository (REP) member selection procedure	MOPSO determines the domination of each repository member every time new members are admitted. New members are admitted every time new non-dominated solutions are found.	M-MOPSO only determines the domination of some necessary members and any redundant solutions are ignored and not admitted into the repository. As shown in the pseudocode in Section 3.3, M-MOPSO did not check for the domination of all repository members everytime new member is admitted. It only compares the dominance between the non-dominated particle in the population (outside of the repository) and the repository members. As soon as it found that the particle is identical to or is dominated by one of the member, it will not compare with the rest of the member. Aside of reducing the computation time, this also ensures that only unique members exist in the repository.
5. Repository (REP) member deletion procedure	MOPSO deletes its repository members based on the density of the grids, where members in the least populated grid have higher probability to be deleted.	M-MOPSO deletes its repository members based on the Euclidean distance in the objective space between each repository member and the latest admitted member. The members nearer to the latest admitted member have a higher probability to be deleted.
6. Leader selection/ population update	In MOPSO, a single new leader is selected in each iteration and replaces the previous leader. The leader is selected from a repository member in the least populated grid. All other population remains the same as in previous iteration.	In M-MOPSO, each individual have their own saturation counter. If they cannot improve to find better solution after a predetermined number of iteration/s, they will be either replaced by randomly selecting a repository member or by selecting a repository member in the least populated grid.

- $\emptyset_1 = 2.05$: parameter 1
- $\emptyset_2 = 2.05$: parameter 2
- $\emptyset = \emptyset_1 + \emptyset_2$: parameter 3
- $w = \frac{2}{\emptyset - 2 + \sqrt{\emptyset^2 - 4\emptyset}}$: inertia weight
- $wdamp = 0.99$: inertia weight damping rate
- $c_1 = w * \emptyset_1$: personal learning coefficient
- $c_2 = w * \emptyset_2$: global learning coefficient
- $\mu = 0.5$: mutation Rate

The parameters used for MOEA/D are as follows:

- $N = 100$: subproblems (population size)
- $T = 10$: number of neighbours (niche size)
- $n_r = 1$: the maximal copies of a new child in update
- $\delta = 0.9$: the probability of selecting parents from the neighbourhood
- $F = 0.5$: mutation rates
- $CR = 0.5$: crossover rates
- $\eta = 30$: distribution index

The parameters used for MODE are as follows:

- $\varepsilon_{sc} = 0.5$: scaling factor
- $P_m = 0.2$: crossover probability

As MOPSO, MOGWO and M-MOPSO use the same external repository system, they all use the same following parameters:

- $nGrid = 10$: number of grids per dimension
- $nRep = number\ of\ search\ agents$: repository size
- $\alpha = 0.1$: grid inflation rate
- $\beta = 4$: leader selection pressure
- $\gamma = 2$: deletion selection pressure

The performance metric used for comparison is the Inverted Generational Distance (IGD) proposed in [22] which is used for measuring convergence and spread. IGD is chosen as the performance metric in this test because the reference sets are readily available for all benchmark problems. It is used to measure the approximate distance from the Pareto front to the obtained solution set in the objective space. This is achieved by using a set of reference points to represent the true or optimal Pareto front in a given problem. The average distance from each reference point to the nearest obtained solution is calculated to get the IGD value. Thus, the algorithm with the lowest IGD has the best convergence to the optimal Pareto front. IGD is chosen as the performance metric in this test because the reference sets are readily available for all benchmark problems. The mathematical formulation for IGD is as follows:

$$IGD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{22}$$

where n is the number of true Pareto optimal solutions and d_i denotes the Euclidean distance between the i th true Pareto optimal solution in the reference set and the closest obtained Pareto optimal solutions by the algorithm in the objective space:

$$d(a, b) = d(b, a) = \sqrt{\sum_{i=1}^{n} (f_{ia} - f_{ib})^2} \tag{23}$$

where $a = (f_{1a}, f_{2a}, f_{3a}, \ldots, f_{na})$ and $b = (f_{1b}, f_{2b}, f_{3b}, \ldots, f_{nb})$ represent two points in the objective space. Fig. 3 shows an illustration of the IGD metric in two dimensional space.

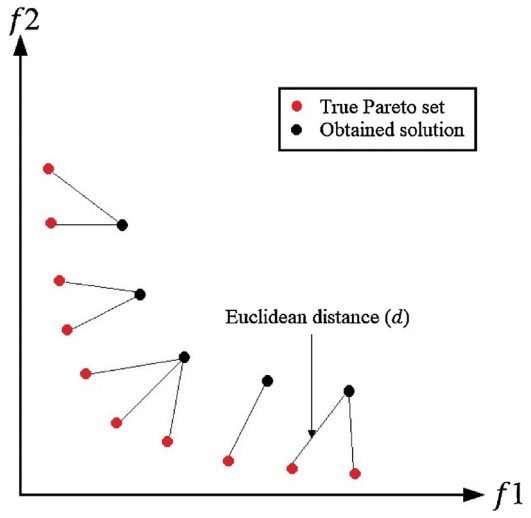


Fig. 3. Illustration of the IGD metric.

Table 2
Distance parameter test.

Distance	CF5 (bi-objective)			
	0 (No Shift)	0.01	0.1	0.2
Mean	0.24964	0.22935	**0.20899**	0.22159
Median	0.20576	0.19565	0.19251	0.18957
Std. Dev.	0.12533	0.09692	0.07403	0.11081
Max	0.58576	0.60154	0.40945	0.59269
	CF7 (bi-objective)			
Mean	0.18439	0.19755	**0.16155**	0.20978
Median	0.19078	0.19749	0.14967	0.19254
Std. Dev.	0.08996	0.07805	0.06373	0.11564
Max	0.44043	0.40582	0.29748	0.47323
	CF10 (tri-objective)			
Mean	1.60301	1.14853	**0.98001**	1.29067
Median	0.53475	0.50492	0.47228	0.46505
Std. Dev.	1.87088	1.52452	1.39953	1.76438
Max	6.24084	5.43766	5.03013	5.79894

Bold values signify the best values obtained among all the tested algorithms, either minimum or maximum.

In our experiment, we used 1000 evenly distributed solutions as the reference set for test instances with two objectives except for CF1 which used 21 solutions, while 10,000 solutions are used for test instances with three objectives, which are CF8, CF9 and CF10.

In order to verify their performance in solving constraint problems, the algorithms are run 30 times for each test problem. Statistical analysis are provided in Table 9 and Fig. 6. To represent the qualitative results, the Pareto optimal solutions obtained by each algorithm on the objective space are provided in Figs. 7 and 8.

Before comparing M-MOPSO with other algorithms, we did three tests to find the best parameters for M-MOPSO. In each test, the algorithm is run 30 times for three different test instances, namely CF5, CF7, and CF10. The first test is to find the best distance between the solution variables and the problem boundary. In Section 3.1, we discussed the boundary control mechanism for M-MOPSO and mentioned that solution variables which are close to their respective problem boundary will be replaced by the value of the boundary. In order to determine the effectiveness of this method, we run a preliminary test to find the best distance to be use in Eq. (15). Table 2 shows M-MOPSO's performance with different distance parameter. After testing with three different test instances, we found that 0.1 is the best value. The second test is to find the parameter value for S_{f1}. Based on Table 3, bi-objective problems solved by M-MOPSO required S_{f1} value of 10 as best performing parameter value whereas solving tri-objective problems required S_{f1} to be 15 to merit best optimization performance. In the third test given in Table 4, S_{f2} set to 0.97 and 0.99 performed better than other values for bi-objective problems and tri-objectives problem respectively. Based on these results, we determined the best parameters for M-MOPSO, which are given in Sections 3.1 and 4.1. The same set of parameters are used throughout the rest of experiments in this paper for the sake of consistency.

Table 3
S_{f1} parameter test.

S_{f1}	CF5 (bi-objective)		
	5	10	15
Mean	0.25283	**0.20899**	0.21496
Median	0.20497	0.19251	0.16637
Std. Dev.	0.13223	0.07403	0.09578
Max	0.57302	0.40945	0.51629
	CF7 (bi-objective)		
Mean	0.18011	**0.16155**	0.17446
Median	0.17671	0.14967	0.15851
Std. Dev.	0.07788	0.06373	0.09556
Max	0.43995	0.29748	0.46509
	CF10 (tri-objective)		
Mean	0.95236	0.98001	**0.75825**
Median	0.46549	0.47228	0.50045
Std. Dev.	1.31248	1.39953	1.03175
Max	4.59647	5.03013	4.92504

Bold values signify the best values obtained among all the tested algorithms, either minimum or maximum.

Table 4
S_{f2} parameter test.

S_{f2}	CF5 (bi-objective)		
	0.95	0.97	0.99
Mean	0.25484	**0.20899**	0.25870
Median	0.20687	0.19251	0.26208
Std. Dev.	0.11930	0.07403	0.10898
Max	0.61713	0.40945	0.57784
	CF7 (bi-objective)		
Mean	0.16734	**0.16155**	0.21306
Median	0.13815	0.14967	0.19968
Std. Dev.	0.08392	0.06373	0.11302
Max	0.49474	0.29748	0.48598
	CF10 (tri-objective)		
Mean	1.34029	0.98001	**0.47143**
Median	0.54038	0.47228	0.48625
Std. Dev.	1.63142	1.39953	0.09777
Max	4.84263	5.03013	0.65184

Bold values signify the best values obtained among all the tested algorithms, either minimum or maximum.

4.2. Fed-batch bioprocess problems

Two case studies will be used for our study in multi-objective bioprocess problems.

4.2.1. Case study I

In this case study, we will address the lysine fermentation model proposed by Ohno et al. [23]. The model equations are as follows:

$$\frac{dx_1}{dt} = \mu x_1 \tag{24}$$

$$\frac{dx_2}{dt} = FS_F - \sigma x_1 \tag{25}$$

$$\frac{dx_3}{dt} = \pi x_1 \tag{26}$$

Table 5
Variables definitions for case study I.

State variables	Definitions
x_1	Cell mass (g/L)
x_2	Substrate concentrations (g/L)
x_3	Product (Lysine) concentrations (g/L)
x_4	Fermenter volume (L)
F	Substrate volumetric feeding rate (L/h)
S_F	Substrate feed concentration (g/L)
μ	Specific growth rates
σ	Substrate consumption
π	Product formation

Table 6
Parameter values for case study I.

Parameter	Value
$x_1(0)$	0.1 g/L
$x_2(0)$	14 g/L
$x_3(0)$	0 g/L
$x_4(0)$	5 L
S_F	2.8 wt%

$$\frac{dx_4}{dt} = F \tag{27}$$

where

$$\mu = 0.125x_2 \tag{28}$$

$$\sigma = \frac{\mu}{0.135} \tag{29}$$

$$\pi = -384\mu^2 + 134\mu \tag{30}$$

Table 5 presents the variables used in case study I. The constraints used are: $x_4(t) \leq 20$ and $0 \leq F(t) \leq 2$. In Table 6, the initial state conditions and the value of S_F are given.

There are two performance indexes (PI) which are needed to be maximised. The first PI is the productivity (J_p) while the second PI is the yield (J_y). These are defined as follows:

$$J_p = \frac{x_3(t_f)}{t_f} \tag{31}$$

$$J_y = \frac{x_3(t_f)}{\int_0^{t_f} F(t)S_F dt} \tag{32}$$

where the final time, t_f is an additional variable to be found by the algorithm within the range of 30–40 h. The number of intervals for the feeding sequence is set as 20 intervals.

4.2.2. Case study II

The production of induced foreign protein by recombinant bacteria proposed by Lee and Ramirez is investigated [24] incorporating parameter function set with improved control sensitivity by Tholudur and Ramirez [25]. The model equations [25] are as follows:

$$\frac{dx_1}{dt} = u_1 - u_2 \tag{33}$$

$$\frac{dx_2}{dt} = g_1 x_2 - \frac{u_1 + u_2}{x_1} x_2 \tag{34}$$

$$\frac{dx_3}{dt} = \frac{100u_1}{x_1} - \frac{u_1 + u_2}{x_1} x_3 - \frac{g_1}{0.51} x_2 \tag{35}$$

$$\frac{dx_4}{dt} = R_{fp} x_2 - \frac{u_1 + u_2}{x_1} x_4 \tag{36}$$

$$\frac{dx_5}{dt} = \frac{4u_2}{x_1} - \frac{u_1 + u_2}{x_1} x_5 \tag{37}$$

Table 7
Variables used for case study II.

State variables	Definitions
x_1	Volume of reactor (L)
x_2	Concentrations of cell (g/L)
x_3	Concentrations of Substrate (g/L)
x_4	Concentrations of foreign protein (g/L)
x_5	Concentrations of inducer (g/L)
x_6	Shock factors of inducer on the cell growth rate
x_7	Factors of recovery on the cell growth rate
u_1	Feed rates of glucose (L/h)
u_2	Feed rates of inducer (L/h)

Table 8
Parameter values for case study II.

Parameter	Value
t_f	10 hours
$x_1(0)$	1 L
$x_2(0)$	0.1 g/L
$x_3(0)$	40 g/L
$x_4(0)$	0 g/L
$x_5(0)$	0 g/L
$x_6(0)$	1 g/L
$x_7(0)$	0 g/L

$$\frac{dx_6}{dt} = -k_1 x_6 \tag{38}$$

$$\frac{dx_7}{dt} = k_2(1 - x_7) \tag{39}$$

The process kinetics are given by:

$$g_1 = \left(\frac{x_3}{14.35 + x_3(1 + \frac{x_3}{111.5})}\right)\left(x_6 + \frac{0.22x_7}{0.22 + x_5}\right) \tag{40}$$

$$R_{fp} = \left(\frac{0.233x_3}{14.35 + x_3(1 + \frac{x_3}{111.5})}\right)\left(\frac{0.005 + x_5}{0.022 + x_5}\right) \tag{41}$$

$$k_1 = k_2 = \frac{0.09x_5}{0.034 + x_5} \tag{42}$$

The PI is defined as:

$$\underset{u_1(t),\, u_2(t)}{Maximize}\ J_1 = x_4\left(t_f\right) x_1\left(t_f\right). \tag{43}$$

$$\underset{u_1(t),\, u_2(t)}{Minimize}\ J_2 = \int_0^{t_f} u_2(t)\, dt. \tag{44}$$

Table 7 gives the detail of variables used for case study II. The constraint for inputs are: $0 \leq u_{1,2}(t) \leq 1$. The initial state conditions and final time, t_f are given in Table 8.

The process optimization in fed-batch fermentation requires the determination of the trajectory of its input variables. This can be found by the algorithms in the form of $A \times (B + 1)$ real valued vectors solutions. A is the number of input variables while B stands for the number of feeding intervals. These vectors create a temporal sequence of values in the form of piecewise linear function that consists of B linearly interpolated segments with equal space. In case of multiple input variables presence, all the input variables in A are merged in sequential form to represent solution. In this paper, case study I has one input variable while case study II has two input variables. The value of B for case study I and II are 20 and 25 respectively. The penalty method is employed to handle the constraints.

The solution is used during the numerical simulations which consist of various differential equation models for evaluation. In this work, the Runge-Kutta order 4–5 technique is chosen as the differential equation solver. The results obtained from simulations

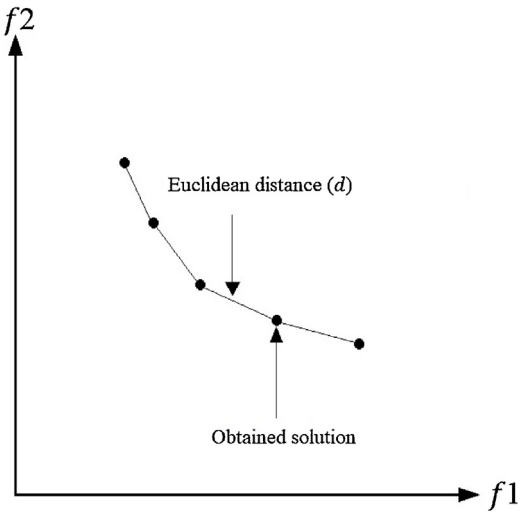


Fig. 4. Illustration of the SP metric.

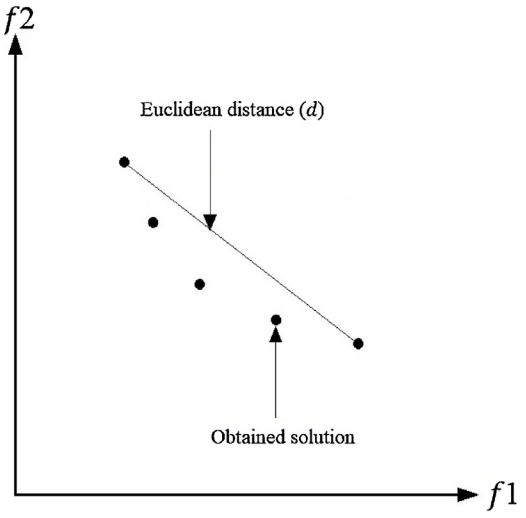


Fig. 5. Illustration of the MS metric.

are then used to calculate the PI and provide the fitness values of the solutions.

For the application problems in case study I and II, 200 search agents are utilized. The algorithms are run for a maximum of 200,000 and 400,000 function evaluations for case study I and II respectively. The same parameters were used by Sarkar and Modak (2005). The Pareto-optimal front between the yield and the productivity for case study I is shown in Fig. 11 while Fig. 12 shows the Pareto-optimal front for Case study II.

In this experiment, we use two performance metrics. The first metric is Spacing (SP) [26], which is used to quantify the coverage by measuring the distance between consecutive solutions obtained in the Pareto front. This metric shows the distribution of the obtained solution, with lower SP value denotes more uniform distribution across the obtained Pareto front. The mathematical formulation for SP is as follows:

$$SP = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\bar{d} - d_i)^2} \tag{45}$$

where $\bar{d}$ is the average of all d_i, n is the number of Pareto optimal solutions obtained, and $d_i = \min_j\left(\left|f_1^i(\vec{x}) - f_1^j(\vec{x})\right| + \left|f_2^i(\vec{x}) - f_2^j(\vec{x})\right|\right)$ for all $i, j = 1, 2, 3, \ldots, n$. Fig. 4 shows an illustration of the SP metric in two dimensional space.

The second metric is Maximum Spread (MS) [27], which measures the extent of spread in the obtained solutions in the Pareto front. Higher MS value indicates that the solutions found at the extreme ends of the obtained Pareto front in each objective space are further apart from each other.

$$MS = \sqrt{\sum_{i=1}^{o}\max(d(a_i, b_i))} \tag{46}$$

where d is a function to calculate the Euclidean distance, a_i is the maximum value in the i th objective, b_i is the minimum value in the i th objective and o is the number of objectives. Fig. 5 shows an illustration of the MS metric in two dimensional space.

We did not use IGD as the performance metric in this experiment because there are no reference set available for the problems.

4.3. Tumor treatment problems

In this paper, the mathematical model proposed by De Pillis and Radunskaya [28] is used where it has three important components. The first component is the immune response which is represented by the growth of immune cells. Tumor presence induces the growth of immune cells which works to destroy tumor cells through a kinetic process. The second component represents competition terms such as tumor cells compete with normal cells for available resources, while immune cells and tumor cells behave in predator-prey fashion. The third component is the logistic growth law.

The following system of differential equations describes the above model:

$$\dot{N} = r_2 N(1 - b_2 N) - c_4 TN - a_3 u\, N(0) = N_0 \tag{47}$$

$$\dot{T} = r_1 T(1 - b_1 T) - c_2 IT - c_3 TN - a_2 u\, T(0) = T_0 \tag{48}$$

$$\dot{I} = s + \frac{\rho IT}{\alpha + T} - c_1 IT - d_1 I - a_1 u\, I(0) = I_0 \tag{49}$$

where I denotes the number of immune cells at time t, T represents the number of tumor cells at time t, N stands for the number of normal (host) cells at time t, and u is the control strategy. s is the influx rate which is assumed as a constant ($0 \le s \le 0.5$). r_i and b_i represent the per capita growth rate and reciprocal carrying capacities of the tumor and normal cells. Tumor, normal tissue and immune are represented by $i = 1$, 2 and 3 identity respectively. a_i are the cell death rates ($0 \le a_i \le 0.5$, with $a_3 \le a_1 \le a_2$), c_i are the competition rates, d_i is the mortality rate, α ($0 \le \alpha \le 0.5$) and ρ ($0 \le \rho \le 1$) are related with inverse declivity of immune response and immune response rate respectively.

The methodology that we use in this problem is the same as the methodology proposed by Lobato, Machado and Steffen [29]. The difference is that we treat the differential equations with the following additional constraints:

$$N \ge 0.75 \tag{50}$$

$$T \ge 0 \tag{51}$$

These constraints are added because they were considered in the original paper by De Pillis and Radunskaya [28].

Furthermore, Lobato, Machado and Steffen [29] solved the problem by combining optimal control theory with multi-objective optimization using metaheuristics, whereas we only use multi-objective optimization metaheuristic method. In our methodology, both the schedule (timing) of medicine input and the amount of

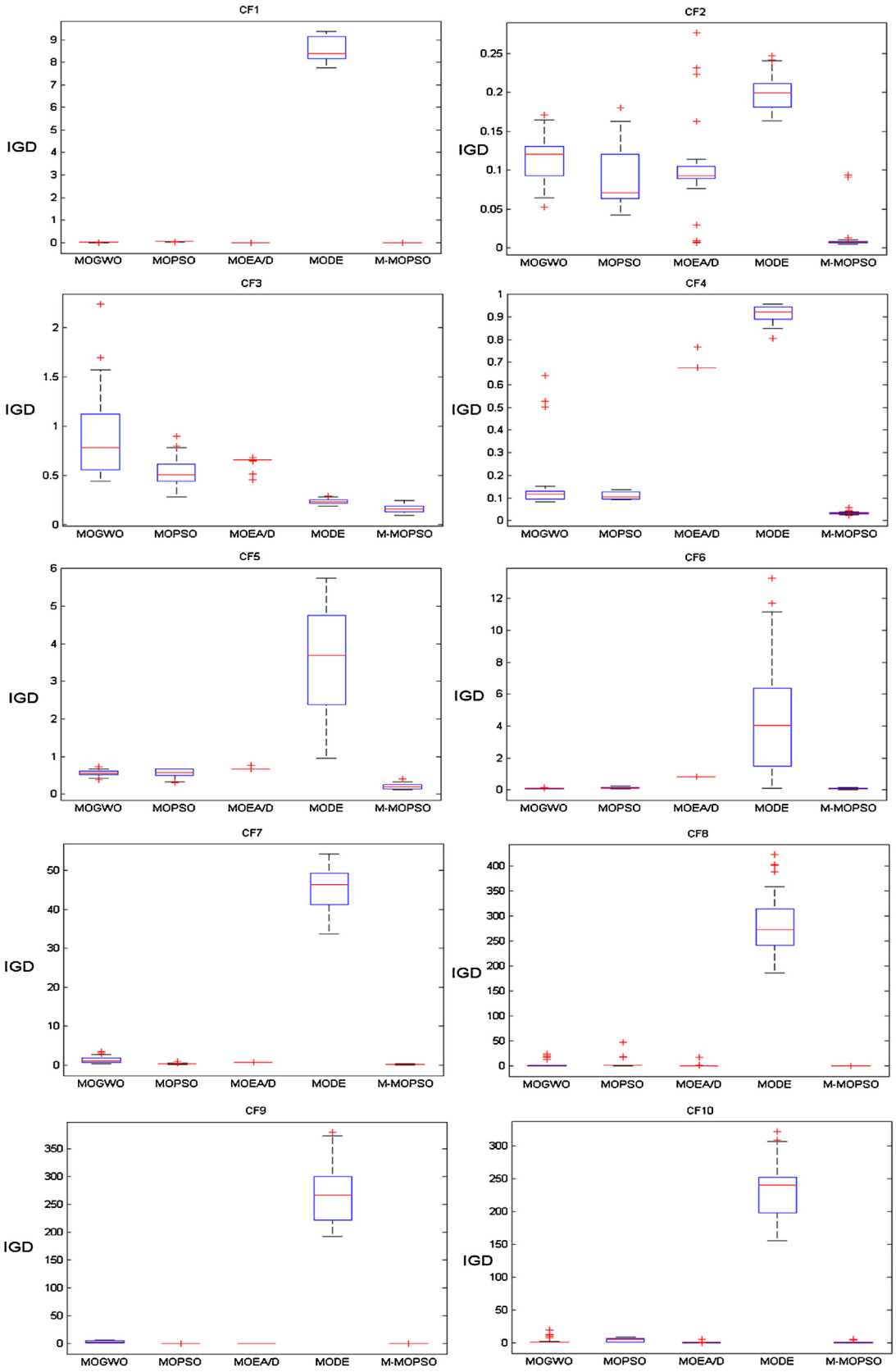


Fig. 6. Boxplot for the statistical results for IGD on CF1–CF10.

medicine (u_{max} or u_{min}) at the particular time are determined by the stochastic algorithm. The schedule is defined as discretized time intervals between the start day of the treatment and the final day $t_i \in [t_0, \ldots, t_f]$, $i = 1, 2, \ldots, Nt$, where Nt is a user-defined maximum time interval. The amount of medicine used in each subinterval follows the bang-bang control and is represented as Nt

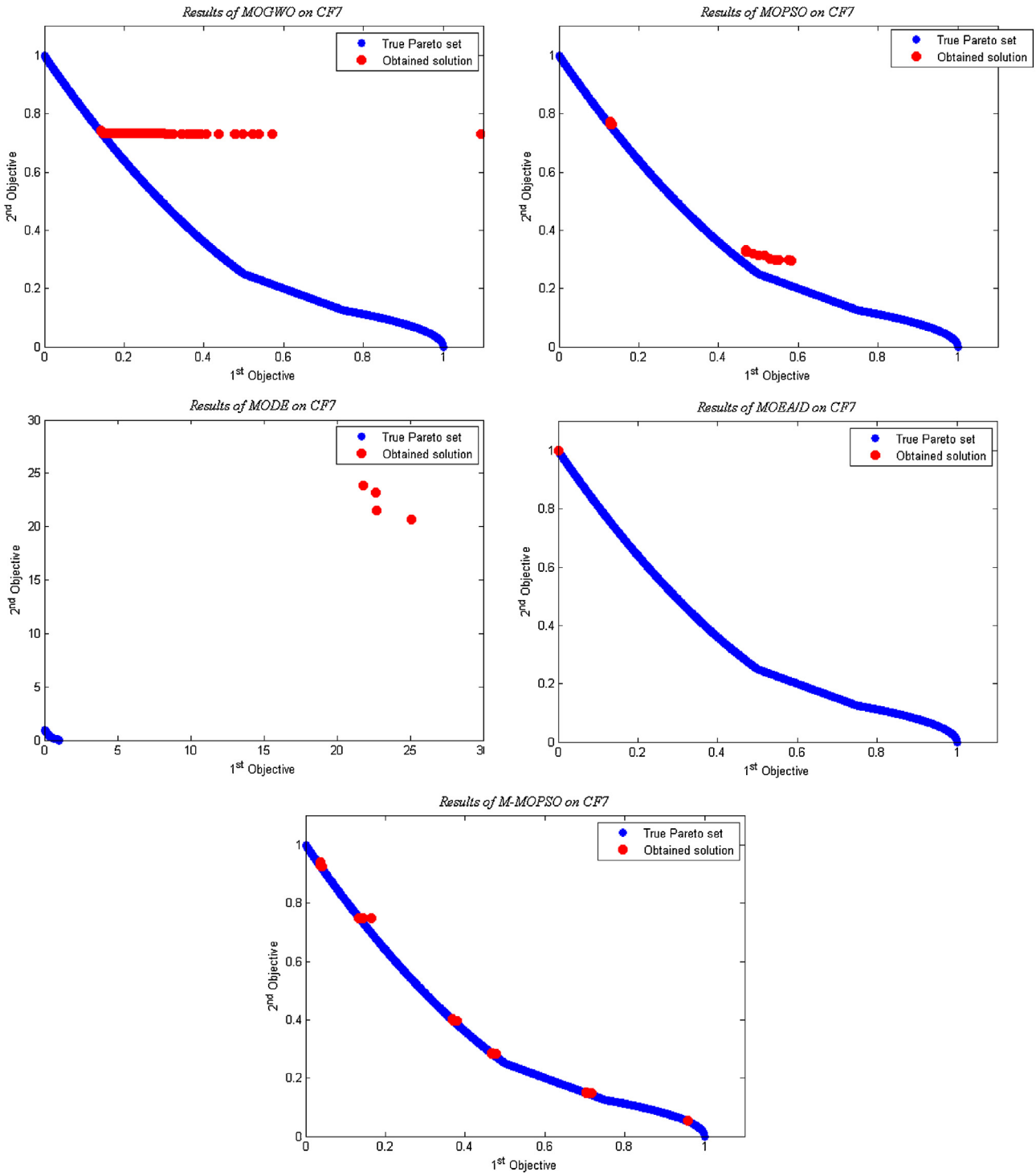


Fig. 7. Obtained Pareto solutions for CF7.

unknown parameters $u_1, u_2, \ldots, u_{Nt}$. Therefore, the total design variables to be optimized by the stochastic algorithm is $2Nt - 1$.

The experiment in this work involves two objectives namely to minimize the concentration of cancer cell and the drug volume given to the patient, defined respectively as:

$$\min \int Tdt$$

$$\min \int udt$$

The control variable is discretized into 15 elements (Nt) and all the differential equations shown in Eqs. (47)–(49) are integrated by Runge–Kutta Method 4–5th order. The initial state conditions are set as follows: $N(0) = 0.9$, $T(0) = 0.25$ and $I(0) = 0.25$. We consider three case studies for tumor treatment problem. The first case study, which we regard as case study III in this paper, uses the following parameters: $a_1 = 0.2, a_2 = 0.3, a_3 = 0.1, b_1 = b_2 = 1, c_1 = c_3 = c_4 = 1, c_2 = 0.5, d_1 = 0.2, r_1 = 1.5, r_2 = 1, s = 0.33, \alpha = 0.3$ and $\rho = 0.01$. For case study IV and case study V, the parameters are the same with case study III except for the value of $s\,(0.30)$ in study IV and the

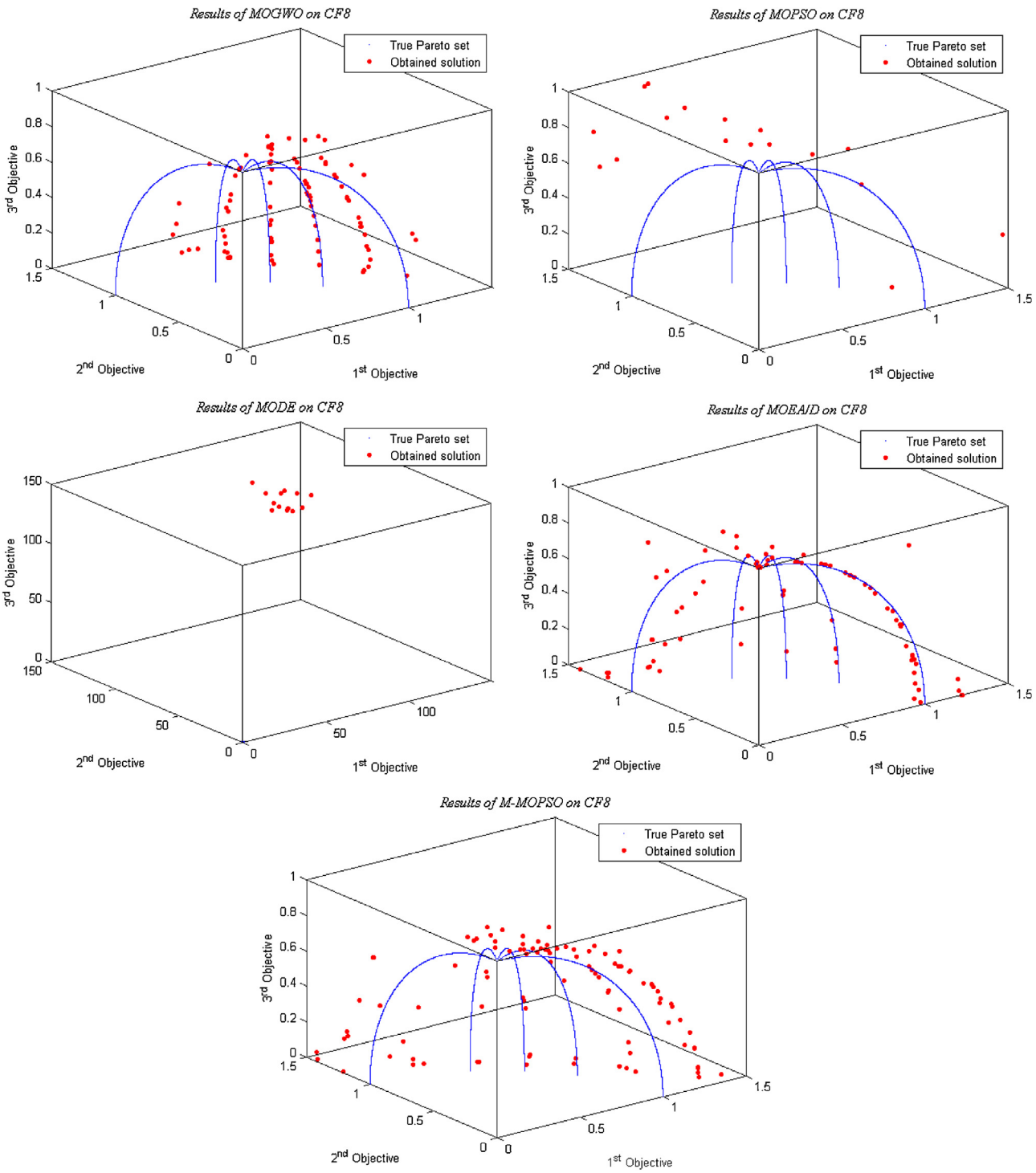


Fig. 8. Obtained Pareto solutions for CF8.

value of $\rho(0.02)$ in case study V. For all algorithms, the number of search agents is 50, the maximum number of iterations is 400 and the maximum number of function evaluations is 20,000. Other specific parameters for each algorithm are defined in Section 4.1.

5. Results and discussion

5.1. Benchmark problems

Based on Table 9, M-MOPSO showed improved average results in comparison to all other algorithms across all benchmark problems except for problem CF1 and CF10. MOEA/D obtained the best average for CF1 and CF10. It is worth noting that for CF1 and CF10, M-MOPSO was the second best algorithm after MOEA/D in terms of average result. This shows that in overall, M-MOPSO has good convergence for constrained multi-objective problems and rivals other algorithms that we compared against in these experiments.

CF1 features many discontinuous solutions in the Pareto front. Hence, the implementation of a repository mechanism greatly improves the results. The lack of repository system in MODE negatively impacted its performance. MOEA/D performed well in this test problem due to the weighting scheme. M-MOPSO, MOPSO and MOGWO also performed well due to the implementation of

Table 9
IGD results for constrained CEC 2009 benchmark problems.

IGD	CF1 (bi-objective)					CF2 (bi-objective)				
	MOGWO	MOPSO	MOEA/D	MODE	M-MOPSO	MOGWO	MOPSO	MOEA/D	MODE	M-MOPSO
Average	0.01284	0.05009	**0.00342**	8.56482	0.00555	0.11466	0.08856	0.10155	0.19962	**0.01319**
Median	0.01337	0.04838	**0.00326**	8.36916	0.00562	0.12056	0.07102	0.09299	0.19901	**0.00718**
STD. Dev.	0.00276	0.00755	0.00132	0.51661	**0.00060**	0.02889	0.03743	0.05864	0.02332	**0.02158**
Worst	0.01651	0.06521	0.00717	9.35663	**0.00700**	0.17114	0.17985	0.27604	0.24734	**0.09347**
	CF3 (bi-objective)					CF4 (bi-objective)				
Average	0.89932	0.52639	0.64998	0.23541	**0.16287**	0.15811	0.11103	0.69385	0.91112	**0.03307**
Median	0.78064	0.51161	0.66162	0.23434	**0.15667**	0.11689	0.10641	0.67518	0.92238	**0.03156**
STD. Dev.	0.43240	0.16151	0.04619	0.02519	**0.04166**	0.13796	0.01516	0.03798	0.03850	**0.00659**
Worst	2.24078	0.89686	0.67927	0.28927	**0.24743**	0.64263	0.13524	0.76853	0.95548	**0.05787**
	CF5 (bi-objective)					CF6 (bi-objective)				
Average	0.55472	0.55721	0.68451	3.54554	**0.20899**	0.08492	0.13105	0.81639	4.30615	**0.06520**
Median	0.56186	0.57480	0.67518	3.69688	**0.19251**	0.08118	0.12431	0.81639	4.02548	**0.05716**
STD. Dev.	0.08743	0.11383	0.02848	1.35938	**0.07403**	0.02194	0.03990	**0.00000**	3.60753	0.02877
Worst	0.72049	0.67521	0.76853	5.73554	**0.40945**	0.16408	0.21753	0.81639	13.2606	**0.12662**
	CF7 (bi-objective)					CF8 (tri-objective)				
Average	1.39173	0.37812	0.81639	45.8691	**0.16155**	4.82411	4.29267	0.77133	283.612	**0.19009**
Median	1.07876	0.33577	0.81639	46.3603	**0.14967**	0.61483	0.50137	**0.14983**	272.578	0.18861
STD. Dev.	0.89057	0.15463	0.00000	**5.5944**	0.06373	8.00306	9.97090	3.08474	65.151	**0.01965**
Worst	3.41388	0.95016	0.81639	54.1519	**0.29748**	23.64818	47.3373	17.07868	423.237	**0.24888**
	CF9 (tri-objective)					CF10 (tri-objective)				
Average	2.48865	0.26847	0.13579	268.447	**0.11914**	3.14974	4.36486	**0.59738**	231.203	0.98001
Median	2.55997	0.27029	0.13789	266.811	**0.11640**	1.01020	5.42038	**0.45683**	240.133	0.47228
STD. Dev.	1.95641	0.03127	0.02274	51.116	**0.01724**	4.71217	2.96877	**0.94537**	43.123	1.39953
Worst	5.78166	0.32490	0.17516	378.677	**0.16263**	19.36988	8.57767	5.56766	321.288	**5.03013**

Bold values signify the best values obtained among all the tested algorithms, either minimum or maximum.

repository system. CF2 is a convex function with two disconnected segments in its Pareto front. CF3 is a concave function with two disjointed segments in its Pareto front. In both CF2 and CF3, M-MOPSO converged better than other algorithms. CF4 and CF5 have identical Pareto shape, as well as CF6 and CF7. CF8, CF9 and CF10 are three objective problems.

Fig. 6 shows the boxplot for IGD results which illustrates the data distribution. Based on Fig. 6, M-MOPSO shows clear improvement compared to other algorithms for CF2, CF3, CF4 and CF5. It obtained better median IGD and more consistent result as indicated by the lower and shorter boxplot compared to others. For other benchmark problems, the distinction between each algorithm is difficult to observe due to the noticeably poor performance of MODE compared to others.

Figs. 7 and 8 illustrate both the coverage and convergence of the best solution found by each algorithm for some of the test instances. The higher proximity of solutions to the true Pareto front reveals the improved convergence of the solutions obtained. For CF7, M-MOPSO obtained a more distributed solution which all converged on Pareto front compared to others, as shown in Fig. 7. In Fig. 8, there are five disconnected parts which represent the Pareto front of CF8 in three dimensional objective space. M-MOPSO obtained solutions that cover almost all parts. Overall, M-MOPSO is able to obtain more solutions with closer proximity to the true optimal Pareto fronts of each test functions compared to other algorithms.

The convergence graph for CF7 is shown in Fig. 9. The graph shows the average Inverted Generational Distance, IGD of 30 runs recorded at 30 intervals of function evaluations (FE) obtained by each algorithm. MOEA/D has the best IGD at 10,000 FE but rapidly saturates at 20,000 FE. M-MOPSO however, matches the IGD of MOEA/D at 20,000 FE and saturates at lowest IGD among all the algorithms at 30,000 FE.

In Fig. 10, the convergence graph of three objective problem, CF8 is shown. At 10,000 and 20,000 FE, both M-MOPSO and MOGWO have almost the same IGD. However, at 30,000 FE, the IGD of MOGWO starts to become worse while the IGD of M-MOPSO continues to improve. This scenario remains until termination.

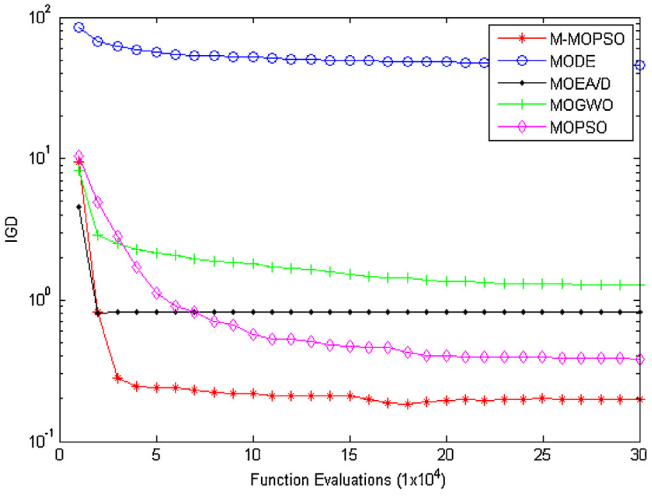


Fig. 9. Convergence graph for CF7.

Overall, M-MOPSO shows significant improvement at 30,000 FE in comparison with all other algorithms for the constrained problem of CF5, CF7 and CF8.

5.2. Fed-batch bioprocess problems

In our experiments, we applied M-MOPSO to address multi-objective fed-batch problems. Two case studies (case study I and II) were investigated and the results were compared with MOPSO, MOEA/D, MODE and MOGWO. Fig. 11 shows that MOPSO is not effective in solving the constraint problem of case study I and could only find one feasible solution. M-MOPSO however, obtained a very good Pareto front and rivalled the performance of MOEA/D. In terms of the maximization of the Pareto front in both axes, M-MOPSO has the best performance in higher yield scale while MOEA/D has the best performance in higher productivity scale.

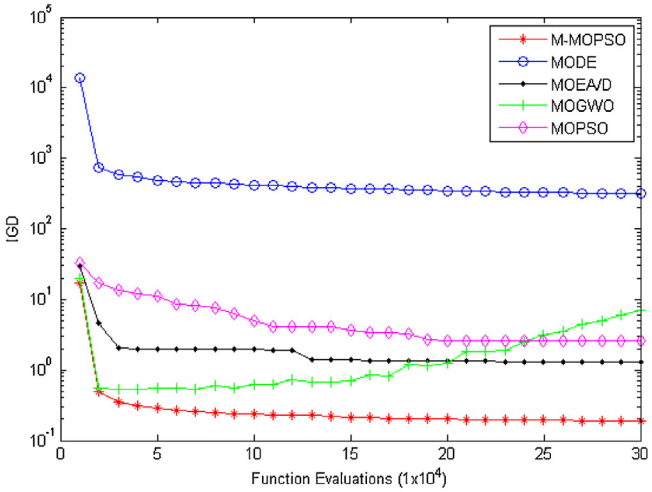


Fig. 10. Convergence graph for CF8.

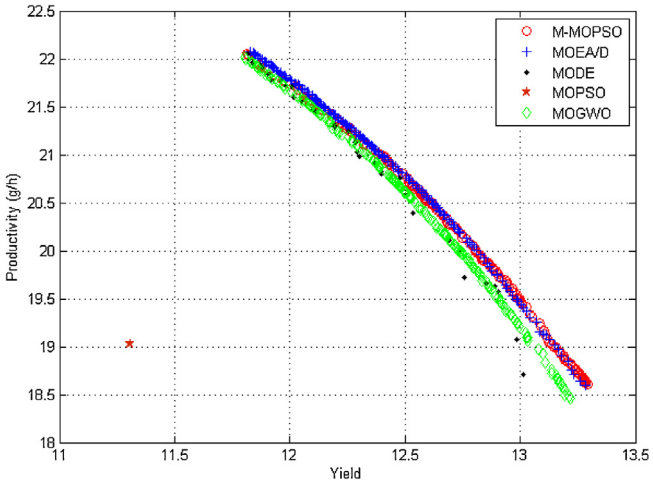


Fig. 11. Productivity-yield Pareto-optimal front for case study I.

In Fig. 12, all algorithms have almost equal convergence for this problem. The difference that set them apart is the spread of the Pareto front. M-MOPSO has the best coverage of the whole objective space. The distribution of the solutions found by M-MOPSO extends towards a greater area compared to other algorithms, which is evident based on MS in Table 10. In terms of coverage, M-MOPSO is followed by MOPSO, MOEA/D, MOGWO and MODE. In general, M-MOPSO obtained a good Pareto front for case study II.

Table 10 shows the results of spacing (SP) and maximum spread (MS) for all algorithms in case study I and II respectively. In case study I, apart from MOPSO which only found one unique solution, M-MOPSO emerged with the lowest SP in comparison to the others. This shows that M-MOPSO has the most uniform distribution of its Pareto front. For MS, M-MOPSO obtained the highest value which

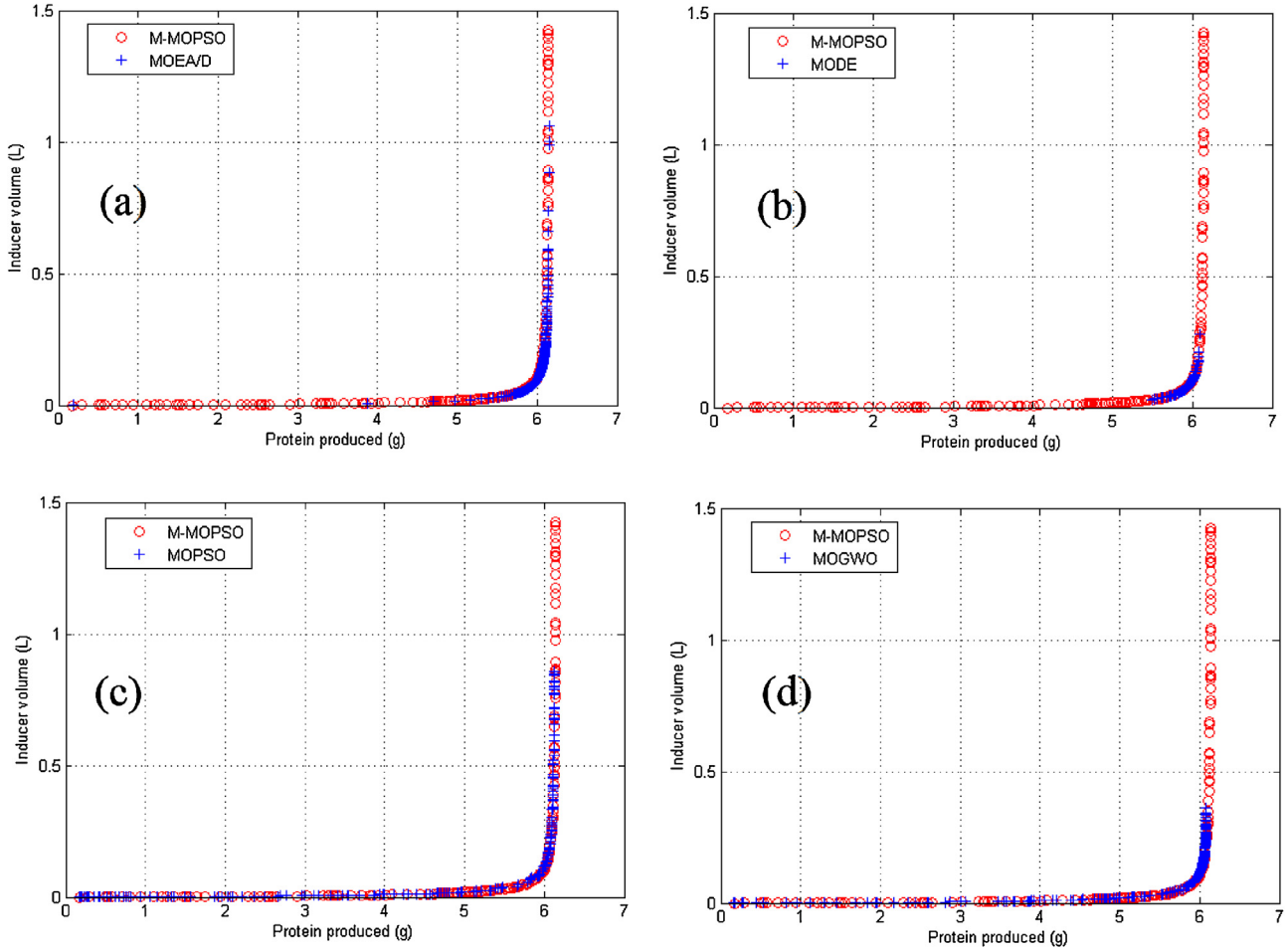


Fig. 12. Pareto-optimal front of M-MOPSO against others for case study II: (a) M-MOPSO against MOEA/D, (b) M-MOPSO against MODE, (c) M-MOPSO against MOPSO, (d) M-MOPSO against MOGWO.

Table 10
SP and MS results for chemical problems.

	Case1		Case2	
	MS	SP	MS	SP
M-MOPSO	**2.22884**	**0.01298**	**2.71857**	**0.0106**
MODE	2.13055	0.09649	0.92574	0.03412
MOEA/D	2.22270	0.01578	2.65010	0.06437
MOGWO	2.21730	0.01379	2.50830	0.03134
MOPSO	–	–	2.60926	0.02751

Bold values signify the best values obtained among all the tested algorithms, either minimum or maximum.

means that it has the largest spread compared to others. The Pareto front for M-MOPSO is also more maximized compared to MOPSO and MOGWO.

M-MOPSO also obtained the lowest SP compared to others in case study II. Subsequently, M-MOPSO obtained the largest MS, hence providing better balance between SP and MS compared to others.

Overall, M-MOPSO was able to solve multi-objective bioprocess application problems effectively. This can be seen by the results obtained in both case study I and II, where M-MOPSO edged over most algorithms tested in this study. It is also worth noting that the time taken for all the algorithms to complete the simulation is around four and twelve hours for case study I and II respectively. This high amount of time taken can be attributed to the usage of Runge-Kutta method for solving the numerous ordinary differential equations.

5.3. Tumor treatment problems

We broadened our investigation on the capabilities of M-MOPSO by addressing tumor treatment problems. Three case studies (case study III, IV and V), which represent three different scenarios in chemotherapy are presented. Fig. 13 shows the Pareto front obtained by the five algorithms for case study III. All algorithms obtained almost similar results. However, M-MOPSO obtained a slightly better spread in its Pareto front. Fig. 14 shows the control variable profile and the cells concentration for case study III obtained by M-MOPSO at the point nearest to (0, 0) along the Pareto front. It can be seen in Fig. 14(a) that the drug is administered till the eight day. After the eighth day, no drug is administered. Hence,

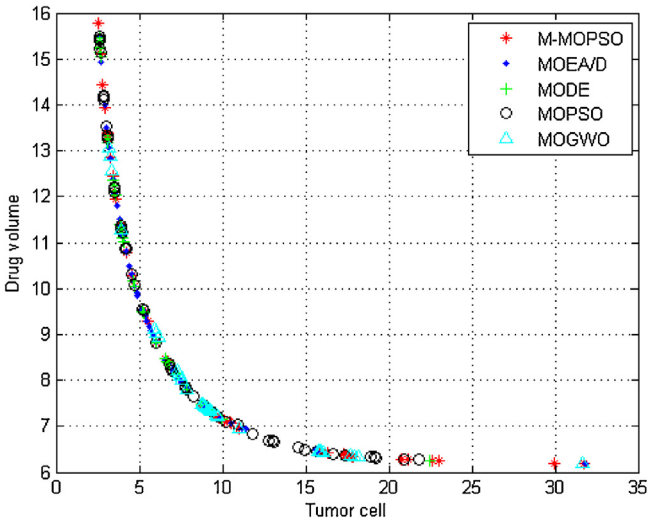
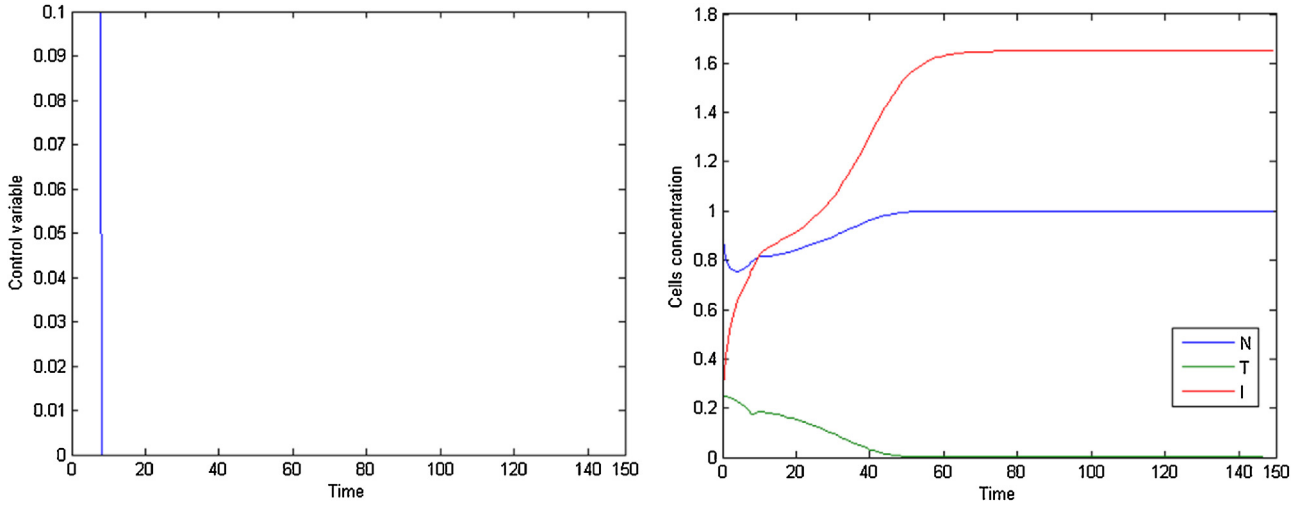


Fig. 13. Pareto optimal front for case study III.

we can see in Fig. 14(b), the sharp decline of both normal cells and tumor cells concentration up until the eighth day. After eight days, the immune cells concentration is high enough to kill the remaining tumor cells.

Fig. 15 shows the Pareto front obtained by the five algorithms for case study IV. M-MOPSO and MOEA/D obtained almost similar Pareto fronts, which are better than the other algorithms. Fig. 16 shows the control variable profile and the cells concentration for case study IV obtained by M-MOPSO at the point nearest to (0, 0) along the Pareto front. This figure shows a similar pattern with case study III except that the period of drug administration is longer, until the fourteenth day. This is due to the lower amount of immune cells in case study IV.

Fig. 17 shows the Pareto front obtained by the five algorithms for case study V. All algorithms obtained almost similar Pareto front. MOGWO's solutions are more focused on the lower portion of the drug volume while MODE's are more focused on the upper portion of the drug volume. MOEA/D is more focused on the centre of the Pareto front while MOPSO's and M-MOPSO's solutions are more evenly spaced throughout the Pareto front. Fig. 18 shows the control variable profile and the cells concentration for case study V



(a) Control variable profile (b) Cells concentration

Fig. 14. Control variable and cells concentration for case study III.

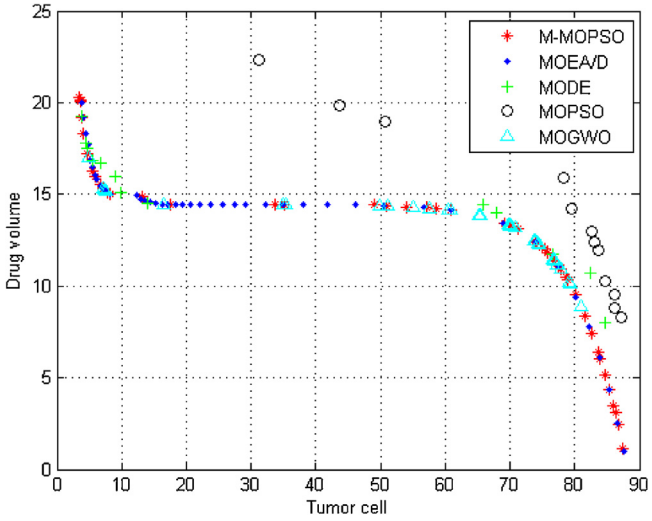


Fig. 15. Pareto optimal front for case study IV.

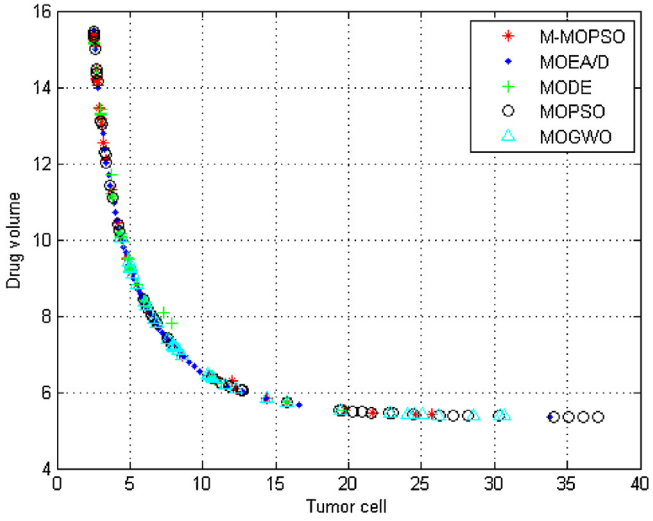


Fig. 17. Pareto optimal front for case study V.

obtained by M-MOPSO at the point nearest to (0, 0) along the Pareto front. This figure also shows a similar pattern with case study III except that the period of drug administration is slightly shorter, until the seventh day, due to the higher amount of immune cells.

Tables 11, 13 and 15 show the minimum Euclidean distance between the Pareto front obtained by each algorithm and the point (0, 0); the tumor cell and drug volume at the corresponding point; and the time taken by each algorithm in case study III, IV and V respectively. In case study III, M-MOPSO found the closest solution to the point (0, 0) and the lowest tumor cell concentration. All algorithms except MOGWO found the lowest drug concentration. In case study IV, M-MOPSO obtained the lowest distance and least time taken. MODE has the lowest tumor cell concentration while MOGWO has the lowest drug volume. In case study V, M-MOPSO has the lowest distance, lowest tumor cell concentration and least time taken. MOPSO has the lowest drug volume.

Tables 12, 14 and 16 shows the results of MS and SP for case study III, IV and V respectively. These results are obtained after five runs for each case which provide the minimum, maximum and average for both MS and SP. In case study III, M-MOPSO obtained the best mean MS while MOPSO obtained the best mean SP. However,

Table 11
Results for case study III.

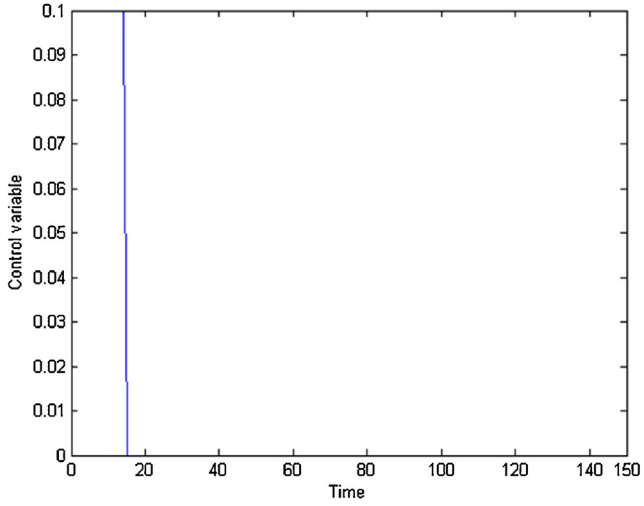
Algorithm	Distance	Tumor cell concentration	Drug volume	Time
M-MOPSO	**10.6777**	5.9964	8.8349	**1626.581**
MOEA/D	10.6813	6.0028	8.8349	1764.92
MODE	10.6835	6.0067	8.8349	1656.01
MOPSO	10.6898	6.0180	8.8349	1734.81
MOGWO	10.8342	6.1341	8.9304	1647.00

Bold values signify the best values obtained among all the tested algorithms, either minimum or maximum.

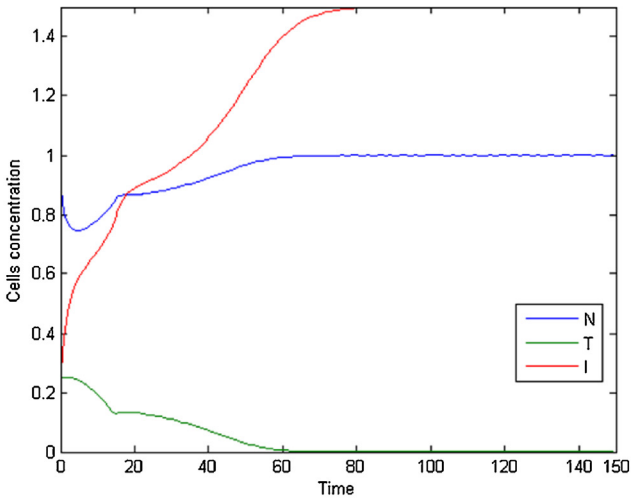
Table 12
MS and SP for case study III.

Algorithm	MS				SP			
	Max	Min	Mean	S.D.	Max	Min	Mean	S.D.
M-MOPSO	**6.5008**	**5.6112**	**6.0626**	**0.421**	1.7801	0.4090	0.8830	0.6072
MOEA/D	6.4761	5.1746	6.0261	0.5016	2.7277	0.8417	1.9598	0.8242
MODE	6.0257	3.8903	4.9471	0.8997	3.2860	0.3073	1.5655	1.2394
MOGWO	6.3430	5.0219	5.8651	0.4995	1.9328	0.3707	0.7465	0.6673
MOPSO	6.4457	1.0675	4.1889	2.3256	**0.6300**	**0.0051**	**0.2768**	**0.2698**

Bold values signify the best values obtained among all the tested algorithms, either minimum or maximum.

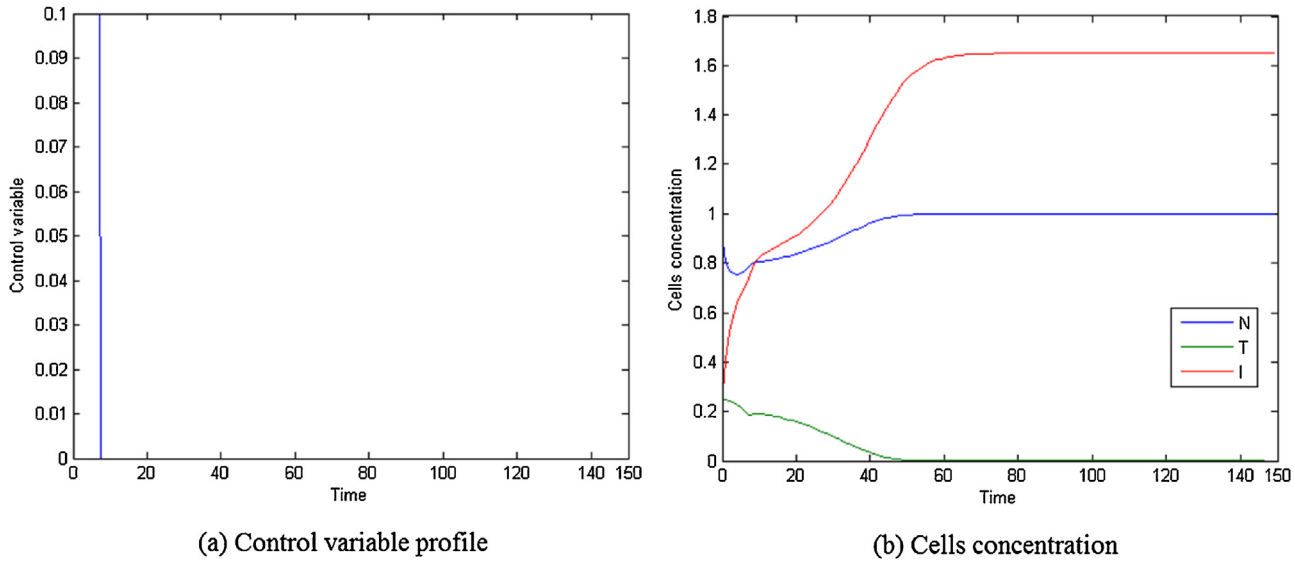


(a) Control variable profile



(b) Cells concentration

Fig. 16. Control variable and cells concentration for case study IV.



(a) Control variable profile (b) Cells concentration

Fig. 18. Control variable and cells concentration for case study V.

Table 13
Results for case study IV.

Algorithm	Distance	Tumor cell concentration	Drug volume	Time
M-MOPSO	**16.890**	6.7024	15.503	**1706.1**
MOEA/D	16.892	6.7646	15.479	1721.8
MODE	17.669	5.4014	16.823	1707.8
MOPSO	38.321	31.130	22.347	1892.2
MOGWO	16.904	7.2054	15.292	1806.1

Bold values signify the best values obtained among all the tested algorithms, either minimum or maximum.

Table 15
Results for case study V.

Algorithm	Distance	Tumor cell concentration	Drug volume	Time
M-MOPSO	**10.274**	5.9957	8.3433	**1512.4**
MOEA/D	10.278	6.1525	8.2328	1542.5
MODE	10.275	6.1698	8.2158	1682.5
MOPSO	10.275	6.1817	8.2076	1677.1
MOGWO	10.285	6.1360	8.2536	1613.6

Bold values signify the best values obtained among all the tested algorithms, either minimum or maximum.

M-MOPSO obtained better balance between MS and SP considering that MOPSO has the lowest MS among all algorithms. This means that M-MOPSO has the best trade-off between coverage and distribution of the Pareto front. In case study IV, M-MOPSO obtained the best mean MS and mean SP. In case study V, M-MOPSO obtained the best mean MS and mean SP. Hence, M-MOPSO has the best balance between the maximum spread of its Pareto front and the spacing of its individual solutions among all the algorithms. Also worth noting is that M-MOPSO obtained relatively low standard deviations for all cases. This shows that M-MOPSO has consistent performance for these problems and can handle constraints effectively. In comparison, the original MOPSO has relatively higher standard deviations for all cases. This is due to its inability to find feasible solutions in some of the runs. Overall, for all cases of tumor treatment problem, M-MOPSO performed effectively by providing solutions with high coverage of the Pareto front, while maintaining consistent performance.

Based on all the results obtained, M-MOPSO is clearly effective in solving multi-objective problems presented. In summary, while some algorithms may excel in some problems but were inferior in other problems, M-MOPSO managed to perform consistently in all problems presented in this paper. It improved upon the original MOPSO in a few areas, especially in the constrained handling capability. Throughout the experiments, in some of the problems, the original MOPSO requires a few reruns to be able to find feasible solutions. M-MOPSO on the other hand, was able to handle constraints effectively and was able to find feasible solutions in a single run. This was partly due to the more robust evolutionary technique which does not rely on only a single leader to search the space but instead each individual set up their own search boundary while evolving in each iteration. In addition, the solutions found by M-MOPSO were more spread out and more evenly spaced compared to other algorithms tested in this study, namely MOEA/D, MODE, MOGWO and MOPSO, as shown in the benchmark problems and case study II. This was due to the improved repository system where the admittance and removal procedure of the repository members are handled more efficiently, in a manner which enable it to preserve the quality of the Pareto front. Finally, M-

Table 14
MS and SP for case study IV.

Algorithm	MS				SP			
	Max	Min	Mean	S.D.	Max	Min	Mean	S.D.
M-MOPSO	**10.165**	**10.136**	**10.154**	**0.0117**	**1.6358**	1.0428	**1.3073**	0.2352
MOEA/D	10.152	10.072	10.124	0.0311	2.0971	1.5824	1.7829	**0.2017**
MODE	9.6962	9.0593	9.5111	0.2632	34.997	1.9949	10.127	13.952
MOGWO	9.2724	8.4054	8.9523	0.3439	4.1139	1.4244	2.0444	1.1602
MOPSO	10.156	8.3724	9.4817	0.9212	4.2736	**0.8750**	1.7891	1.4177

Bold values signify the best values obtained among all the tested algorithms, either minimum or maximum.

Table 16
MS and SP for case study V.

Algorithm	MS				SP			
	Max	Min	Mean	S.D.	Max	Min	Mean	S.D.
M-MOPSO	**6.6885**	4.0308	**6.0559**	1.1391	**0.5952**	0.4552	**0.5165**	**0.0593**
MOEA/D	6.0075	**5.5795**	5.8503	**0.1809**	2.8488	0.4618	2.1963	0.9940
MODE	5.4080	4.2828	4.7355	0.5160	2.2024	0.1644	0.8231	0.8230
MOGWO	6.5412	5.5661	6.0040	0.3801	1.4985	0.2815	0.7714	0.4877
MOPSO	6.6875	0.0000	3.9786	3.3895	3.3363	**0.0000**	0.8770	1.4000

Bold values signify the best values obtained among all the tested algorithms, either minimum or maximum.

MOPSO was able to perform as good as the state-of-art MOEA/D with the same computational cost but lesser time taken, due to the simplicity and efficiency of all the combined procedures used in M-MOPSO.

6. Conclusion

In this paper, we present a modification of multi-objective particle swarm optimization (MOPSO) to tackle multi-objective optimization problems. Our algorithm, called modified multi-objective particle swarm optimization (M-MOPSO) employs a new dynamic search boundary mechanism to properly balance exploration and exploitation during the search procedure. The archiving procedure used in MOPSO was also modified to maintain diversity in the Pareto front while reducing the computational cost of the archive controller.

Our experiment used the CEC2009 multi-objective benchmark problems to verify the performance of our algorithm. Comparisons were made with four other recent algorithms, namely multi-objective grey wolf optimizer (MOGWO), multi-objective particle swarm optimizer (MOPSO) and multi-objective evolutionary algorithm based on decompositions (MOEA/D) and multi-objective differential evolution (MODE). Based on the results, M-MOPSO emerged as the better algorithm by obtaining better Inverted Generational Distance (IGD) average for eight out of the ten test instances.

We also ran some simulations of bioprocess application problems to investigate the capability of M-MOPSO in solving real-world engineering problems. M-MOPSO showed promising results by rivalling other state-of-the-art techniques used in this study. It also displayed better capability in handling constraint compared to MOPSO. For the unconstraint problem, M-MOPSO obtained superior coverage of the Pareto front while maintaining good convergence compared to other algorithms.

The tumor treatment problem provided another application problem for the tested algorithms. Our M-MOPSO emerged as one of the algorithms that provide the best results in comparison to the other tested algorithms.

In summary, M-MOPSO was able to solve multi-objective problems with good convergence and it is interesting to extend the capability of this algorithm to solve many-objective problems and other more complex applications in the future.

Acknowledgement

This research work is supported by University of Malaya Research Grant (UMRG)RG 333-15AFR, Frontier Science Research Cluster.

References

[1] J. Kennedy, R. Eberhart, Particle swarm optimization Piscataway, NJ, IEEE Int. Conf. on Neural Networks (1995) 1942–1948.
[2] J.H. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis With Applications to Biology, Control and Artificial Intelligence, MIT Press, 1992, pp. 228.
[3] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, Adv. Eng. Softw. 69 (2014) 46–61.
[4] C.A.C. Coello, G.T. Pulido, M.S. Lechuga, Handling multiple objectives with particle swarm optimization, IEEE Trans. Evol. Comput. 8 (3) (2004) 256–279.
[5] Q. Zhang, H. Li, MOEA/D: a multiobjective evolutionary algorithm based on decomposition, IEEE Trans. Evol. Comput. 11 (6) (2007) 712–731.
[6] P.K. Tripathi, S. Bandyopadhyay, S.K. Pal, Multi-objective particle swarm optimization with time variant inertia and acceleration coefficients, Inf. Sci. 177 (22) (2007) 5033–5049.
[7] C.K. Goh, K.C. Tan, D.S. Liu, S.C. Chiam, A competitive and cooperative co-evolutionary approach to multi-objective particle swarm optimization algorithm design, Eur. J. Oper. Res. 202 (1) (2010) 42–54.
[8] B. Xue, M. Zhang, W.N. Browne, Particle swarm optimization for feature selection in classification: a multi-objective approach, IEEE Trans. Cybern. 43 (6) (2013) 1656–1671.
[9] S. Cheng, H. Zhan, Z. Shu, An innovative hybrid multi-objective particle swarm optimization with or without constraints handling, Appl. Soft Comput. 47 (Supplement C) (2016) 370–388.
[10] J. Meza, H. Espitia, C. Montenegro, E. Giménez, R. González-Crespo, MOVPSO: vortex multi-objective particle swarm optimization, Appl. Soft Comput. 52 (Supplement C) (2017) 1042–1057.
[11] R.T. Marler, J.S. Arora, Survey of multi-objective optimization methods for engineering, Struct. Multidiscip. Optim. 26 (6) (2004) 369–395.
[12] Q. Fan, W. Wang, X. Yan, Multi-objective differential evolution with performance-metric-based self-adaptive mutation operator for chemical and biochemical dynamic optimization problems, Appl. Soft Comput. 59 (Supplement C) (2017) 33–44.
[13] Q. Jiang, L. Wang, Y. Lin, X. Hei, G. Yu, X. Lu, An efficient multi-objective artificial raindrop algorithm and its application to dynamic optimization problems in chemical processes, Appl. Soft Comput. 58 (Supplement C) (2017) 354–377.
[14] N. Patel, N. Padhiyar, Multi-objective dynamic optimization study of fed-batch bio-reactor, Chem. Eng. Res. Des. 119 (Supplement C) (2017) 160–170.
[15] D. Sarkar, J.M. Modak, Pareto-optimal solutions for multi-objective optimization of fed-batch bioreactors using nondominated sorting genetic algorithm, Chem. Eng. Sci. 60 (2) (2005) 481–492.
[16] M. Engelhart, D. Lebiedz, S. Sager, Optimal control for selected cancer chemotherapy ODE models: a view on the potential of optimal schedules and choice of objective function, Math. Biosci. 229 (1) (2011) 123–134.
[17] H. Moradi, G. Vossoughi, H. Salarieh, Optimal robust control of drug delivery in cancer chemotherapy: a comparison between three control approaches, Comput. Methods Prog. Biomed. 112 (1) (2013) 69–83.
[18] J.C. Panetta, K.R. Fister, Optimal control applied to cell-cycle-specific cancer chemotherapy, SIAM J. Appl. Math. 60 (3) (2000) 1059–1072.
[19] V.S. Rozova, A.S. Bratus, Therapy strategy in tumour cells and immune system interaction mathematical model, Appl. Anal. 95 (7) (2016) 1548–1559.
[20] Q. Zhang, A. Zhou, S. Zhao, P.N. Suganthan, W. Liu, S. Tiwari. Multiobjective optimization test instances for the CEC 2009 special session and competition. in University of Essex, Colchester, UK and Nanyang technological University, Singapore. Special session on performance assessment of multi-objective optimization algorithms, technical report (2008).
[21] S. Mirjalili, S. Saremi, S.M. Mirjalili, Ld.S. Coelho, Multi-objective grey wolf optimizer: a novel algorithm for multi-criterion optimization, Expert Syst. Appl. 47 (2016) 106–119.
[22] M.R. Sierra, C.A. Coello Coello, Improving PSO-based multi-objective optimization using crowding, mutation and ϵ-dominance Berlin, Heidelberg, Proceedings of the Evolutionary Multi-Criterion Optimization: Third International Conference (2005) 505–519, Springer Berlin Heidelberg.
[23] H. Ohno, E. Nakanishi, T. Takamatsu, Optimal control of a semibatch fermentation, Biotechnol. Bioeng. 18 (6) (1976) 847–864.
[24] J. Lee, W.F. Ramirez, Optimal fed-batch control of induced foreign protein production by recombinant bacteria, AIChE J. 40 (5) (1994) 899–907.
[25] A. Tholudur, W.F. Ramirez, Obtaining smoother singular arc policies using a modified iterative dynamic programming algorithm, Int. J. Control 68 (5) (1997) 1115–1128.

[26] J.R. Schott, Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization: DTIC Document, Massachusetts Institute of Technology. Department of Aeronautics and Astronautics, 1995.
[27] E. Zitzler, Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland.: TIK-Schriftenreihe Nr. 30, Diss ETH No. 13398, Shaker Verlag, Aachen, Germany, 1999.
[28] L.G. De Pillis, A. Radunskaya, The dynamics of an optimally controlled tumor model: a case study, Math. Comput. Modell. 37 (11) (2003) 1221–1244.
[29] F.S. Lobato, V.S. Machado, V. Steffen Jr, Determination of an optimal control strategy for drug administration in tumor treatment using multi-objective optimization differential evolution, Comput. Methods Prog. Biomed. 131 (2016) 51–61.