Expert Systems With Applications 117 (2019) 166–175

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Multiobjective Hooke–Jeeves algorithm with a stochastic Newton–Raphson-like step-size method

O. Tolga Altinoz[1,*], A. Egemen Yilmaz[1]

Department of Electrical and Electronics Engineering, Ankara University, Turkey

ARTICLE INFO

Article history:
Received 10 January 2018
Revised 24 August 2018
Accepted 13 September 2018
Available online 20 September 2018

MSC:
Hooke–Jeeves
Multiobjective optimization
Newton–Raphson
MOEAD
AGE-II
90C29
90C15
90C99

ABSTRACT

Computational optimization algorithms are focused on the improvement of meta-heuristic algorithms in a way that they can able to handle problems with more than one objective; such improved algorithms are called multiobjective optimization algorithms. As the number of objectives is increased, the complexity of the algorithm is increased with respect to the computational cost. Because classical optimization algorithms follow the direction of descending values by calculating derivations of the function, it is possible to evaluate a classical optimization algorithm as the core of a novel multiobjective optimization algorithm. Among the classical optimization algorithms, in this study, the Hooke–Jeeves (HJ) algorithm is selected as the basis of the proposed multiobjective optimization algorithm, in which members of the proposed population-based HJ algorithm move to the Pareto front by checking two neighborhood solutions at each dimension, with a dynamic distance that is calculated by using the Newton–Raphson-like stochastic step-size method. Unlike various multiobjective optimization algorithms, the performance of the proposed algorithm is greatly dependent on the decision space dimension instead of the number of objectives. As the number of objectives increases without changing the decision dimension, the computational cost almost remains the same. In addition, the proposed algorithm can be applied to single, multiple and many objective optimization problems. In this study, initially, the behaviors of the HJ and proposed multiobjective HJ algorithms are evaluated by theoretical and graphical demonstrations. Next, the performance of the proposed method is evaluated on well-known benchmark problems, and the performance of this algorithm is compared with the Nondominated Sorting Genetic Algorithm-II (NSGA-II) algorithm by using three different metric calculations. Finally, the algorithm is applied to many-objective optimization problems, and the performance of the proposed algorithm is evaluated based on the obtained results.

1. Introduction

It is possible to summarize optimization as finding the best solution among all possible solutions for a problem under given constraints. Computational multiobjective optimization is a branch of optimization theory, where there is more than one objective. There is no optimal single solution for multiobjective problems because it is not possible to obtain a single solution that presents the best solution for all objective values. Therefore, there is a trade-off among the solution candidates. Any of the solution candidates that have the best possible objective value for at least one objective without worsening other objective values are called the Pareto optimum solution; this concept was introduced in 1896 (Merritt, 1898). The set containing all Pareto optimum solutions is called the Pareto Front. In general, multiobjective optimization algorithms are evaluated using the domination principle with respect to the Pareto optimality.

The multiobjective optimization algorithms can be divided into two groups with respect to their inherent behavior for finding the approximate solution set (i.e., preferable Pareto Front). These groups are named classical and heuristic (population-based) algorithms. Classical optimization algorithms evaluate the first and/or second derivative of the optimization problems, such as classification methods (Altinoz & Altinoz, 2017), dynamic programming methods (Zhang, Cui, Zhang, & Luo, 2011), or quadratic convex implementations (Zhang, Yang, Liu, & Zhang, 2013). Even this derivative process could be either analytical or numerical, the computation cost of which greatly depends on the complexity of the problem. Moreover, the performance of these classical multiobjective optimization methods is highly depended on the shape of the decision space. The common property of these methods is that

* Corresponding author.
E-mail addresses: taltinoz@ankara.edu.tr (O.T. Altinoz), aeyilmaz@ankara.edu.tr (A.E. Yilmaz).
[1] All authors contributed equally to this manuscript.

https://doi.org/10.1016/j.eswa.2018.09.033
0957-4174/

all of the calculations are made only for a single solution candidate (not population). Even these algorithms aim to have the solutions converge towards the Pareto Front; these are just the modified versions of the classical single objective optimizations algorithms (e.g., linear, nonlinear, and dynamic programming algorithms (Zhang, Zhang, Yang, & Luo, 2015)).

Given the historical background, the first study showing the heuristic algorithms applied to multiobjective algorithms was the Ph.D. thesis of Rosenberg in 1967 (Rosenberg, 1970). In the next decade, the authors focused on obtaining multiobjective population-based optimization algorithms. Among them, the linear aggregation methods are the basic, yet still preferred method. Even this method belongs to the multiobjective branch; it is just a preprocessing mechanism to convert the multiobjective problem into a single objective one. By using this method, the responses of many objectives are weighted and summed to form a single cost value (Coello & Christiansen, 1998; Eklund & Embrechts, 1999). The other fundamental method is called Lexicographic, which is a more systematic approach, such that, initially, it finds the optimum for just a single objective, and then attempt to find the best value for the next objective, until all the objectives finished (Gacôgne, 1997).The design of "modern" multiobjective optimization algorithms began in the 1990s. The most important algorithms are the Strength Pareto Evolutionary Algorithm (Zitzler, Laumanns, & Thiele, 2001) and the nondominated sort algorithm NSGA-II presented by (Deb, Pratap, Agarwal, & Meyarivan, 2002). After 2006, the researchers were generally focused on the improvements in algorithms and designing of metrics. The important improvement during these years was the introduction of (many)-objective evolutionary based on decomposition MOEAD (Zhang & Li, 2007). In addition, in that timeline, the authors focused on applying metrics on evolutionary algorithms; for example, hypervolume metric indicators were applied to the multiobjective problem (Brockhoff & Zitzler, 2007). Most of these studies are based on the evolutionary algorithms. Alternatively, authors attempted to produce multiobjective versions of optimization algorithms, such as differential evolution ((Alatas, Akin, & Karci, 2008; Sarker & Abbass, 2004))and particle swarm optimization (Elhossini, Areibi, & Dony, 2010; Rahimi-Vahed, Mirghorbani, & Rabbani, 2007). These algorithms are population-based, and they generally depend on the production of redundant population and comparison to the original data, with only the best members among these sets are selected in the algorithm. Therefore, the algorithm requires subprograms and function executions, even for very small decision vector sizes.

Currently, although more robust multiobjective algorithms are still being proposed by researchers, classical algorithms remain the preferred method for solving engineering problems. Therefore, in this paper, a classical method, namely, the Hooke–Jeeves (HJ)algorithm (Hooke & Jeeves, 1961) (the paper (Bazaraa, Sherali, & Shetty, 2006) provides some basics related to the convergence properties of the algorithm) is investigated and converted into a multiobjective optimization algorithm. By this approach, a novel multiobjective optimization algorithm that unites both classical and modern methods is presented. The HJ algorithm is one of the oldest algorithms; it is very easy to understand, is applied as an optimization algorithm, and has been applied to many real-life optimization problems. As an intelligence system, the HJ algorithm has been applied to many areas of industrial and engineering problems. In the network optimization problem, the HJ algorithm was applied for the optimal operation of water source allocation without storage (Léon-Celi, Iglesias-Rey, Martínez-Solano, & Savic, 2018). A HJ-inspired algorithm was integrated into a more detailed flying ad hoc problem to determine the path and location of the UAV (Ghazzai, Feidi, Menouar, & Ammari, 2017). Another path planning problem for nautical navigation is one of the problems for which the HJ algorithm has been applied inside a solution framework (Altinoz, Yanar, Ozguven, Demirdogen, & Yilmaz, 2017). In the optic design field, the HJ algorithm was used to obtain the optimal solution from the inverse model of the reflective polarized component of a rough surface (Zhan, Yang, Liu, Zeng, & Wang, 2018). For obtaining the voltage current characteristics of solar cells, the HJ algorithm was modified and applied as a nonlinear optimization algorithm (Ayala-Mató, Seuret-Jiménez, Escobedo-Alatorre, Vigil-Galán, & Courel, 2017). In addition, in the same study, the HJ algorithm was evaluated as a subroutine of a semi-analytic approach, such as a stability problem (Zhang et al., 2008). In Green Engineering, to control river pollution, chaos (Zhang, Ma, Huang, & Wang, 2010) improving the HJ algorithm is preferred and has been applied successfully (Yang & Li, 2010).

Another application of the HJ algorithm is the evaluation of the HJ algorithm with an additional method. Consequently, because the HJ optimization algorithm, in general, has two common properties(e.g., exploration and exploitation or global search and local search), the HJ algorithm is an efficient local search algorithm; hence, the HJ algorithm is merged with other algorithms to improve their local search properties. The HJ and bat algorithms are combined to allow this hybrid algorithm to be applied to solve nonlinear equations (Yang, Zhang, Zhou, & Peng et al. 2018). Moreover, the HJ algorithm is combined with the stochastic local searcher algorithm for fitness diversity and distribution of the solutions within the population in the multi meme differentia evolution framework (Dominguez-Isidro & Mezura-Montes, 2018). The HJ algorithm has even been applied as a part of the hybrid algorithm to solve mixed integer optimization problems (Liu et al., 2018). A hybrid particle swarm optimization and HJ algorithm was applied to a smart building energy optimization problem (Bamdad, Cholette, Guan, & Bell, 2017) and a wireless sensor network problem (Panag & Dhillon, 2017). A hybrid invasive weed optimization with the HJ algorithm was applied to the parameter estimation problem for a pharmacokinetic model (Deng & Li, 2017).

The proposed multiobjective optimization algorithm is an HJ-oriented and population-based algorithm, and every member is a solution instead of a solution candidate (as in classical methods, where the solutions have a direction and step size). At each step, the members are altered, and the subpopulation is produced. Only the members that are close to the Pareto Front survive to the next iteration. As in the HJ algorithm, the step size is the most important part of the proposed algorithm. Therefore, a Newton–Raphson-like step-size updated rule is proposed. After the behaviors of the HJ and proposed algorithms are discussed theoretically, the performance of the proposed method is evaluated on ten benchmark problems and compared with another multiobjective optimization algorithm (namely, NSGA-II). Next, the proposed algorithm is applied to the benchmark problems with three, five, eight, and ten objectives and compared to the multiobjective optimization algorithms NSGA-II, MOEAD, and AGE-II. The advantages and disadvantages of the proposed algorithm are discussed by using these results and the previous results.

The organization of this paper is as follows. The paper begins with the introduction. Section 2 presents the proposed multiobjective HJ algorithm. Section 3 describes the implementation; the performance of the proposed method is also presented in this section. Finally, the conclusion is given in Section 4.

2. Multiobjective Hooke–Jeeves optimization algorithm

In this section, the multiobjective Hooke–Jeeves (MOHJ) algorithm is proposed. First, the conventional single-objective HJ algorithm is presented in brief, and the behavior of the HJ algorithm is discussed theoretically. Next, the proposed method is proposed and explained.

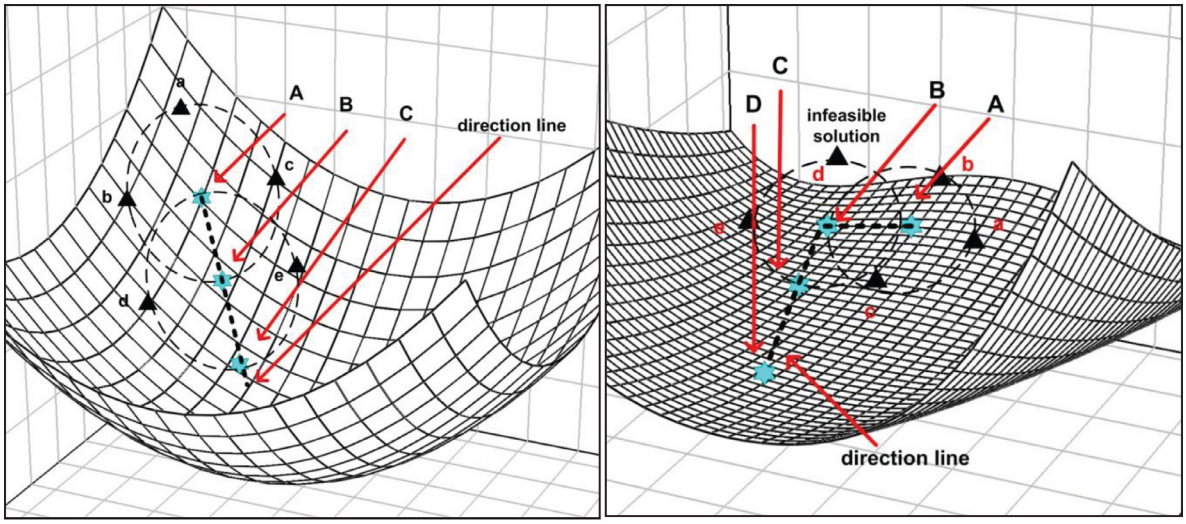

Fig. 1. Behavior of the HJ algorithm on a search space of a) sphere function and b) Rosenbrock function.

2.1. Short review of the Hooke–Jeeves pattern search algorithm

The optimization problems take their name as single objective or multiobjective based on the number of objectives to be considered where multiobjective optimization problems have more than one objective. The general form of the multiobjective optimization problem can be stated as follows:

$$\begin{aligned} &\text{minimize} \quad F(\bar{x}_i) = \{f_1(\bar{x}_i), f_2(\bar{x}_i), ..., f_m(\bar{x}_i)\} \\ &\text{subject to} \quad X = \{\bar{x}_1, \bar{x}_2, ..., \bar{x}_n\} \in \Omega \end{aligned} \tag{1}$$

where Ωrepresents the decision space, such that F: $\Omega \rightarrow R^m$,and $\boldsymbol{x}_i = \{x_{i,1}, x_{i,2}, ..., x_{i,d}\}$ represents the decision variable; d represents the dimension of the decision space, and R^m represents the m dimension of the objective space. There is no single/unique best solution for the multiobjective problems because no single solution can address the best location for all the objectives without the knowledge of the decision maker. Because a trade-off exists among the obtained solution candidates, this trade-off is expressed with the "domination" concept. Let two decision vectors $\boldsymbol{u} = \{u_1, u_2, ..., u_d\}$and $\boldsymbol{v} = \{v_1, v_2, ..., v_d\}$be defined in the decision space; $\boldsymbol{u}, \boldsymbol{v} \in \mathrm{R}^m$. It is said that $\boldsymbol{u}$ weakly dominates $\boldsymbol{v}$ $F(v) \preceq F(u)$ if and only if $f_i(u) \leq f_i(v)$ $\forall i \in \{1, 2, ..., m\}$. It is said that **u** dominates **v** $F(v) \prec F(u)$ if and only if $f_i(u) \leq f_i(v)$, $\forall i \in \{1, 2, ..., m\}$ and $f_j(u) < f_j(v)$, $\exists j \in \{1, 2, 3, ..., m\}$.Among all possible solutions, the non-dominated set is called the Pareto set, the shape that is formed by the Pareto set on the objective space is called the Pareto Front, and any solution in the decision space is called the Pareto optimum solution ($\boldsymbol{x}^*$); all these concepts were introduced in 1896 (Merritt, 1898).

The HJ algorithm is proposed to find the optimum solution of the problem defined in Eq. (1); it begins with the random initial point and the compared local solutions on the search space. These local solutions are at the same distance from the current position. Therefore, this algorithm is also referred to as the pattern search algorithm. At each iteration, two additional solutions are selected on decision space for each dimension. All of these solution candidates are compared with each other. If a better solution is observed, then this point becomes the initial point of the next iteration. This behavior of the algorithm is graphically demonstrated in Fig. 1. In Fig. 1a, it is assumed that a solution candidate begins the iteration from point A. For each dimension, two additional solutions are produced (a,b,c, and B), and their objective functions are compared with respect to the domination principle. Based on their objective function values, the smallest member is the point B, and this point survives to the next iteration. The same comparison is repeated for point B. However, in this case, the distance between additional solutions is changed randomly. This process is repeated until the termination conditions are met. In Fig. 1b, the same behavior is observed for the Rosenbrock function. However, this time, a broken line is observed as the direction line. Moreover, it is possible to encounter the problem that a solution candidate may be outside of the feasible region. In that case, that solution will not be taken into the calculations. The algorithm evaluates two additional solution candidates at each iteration. Hence, for the case that N is the maximum number of iterations (assuming that the global optimal point is reached after N number of iterations for all of the solution candidates are feasible) and D is the dimension of the decision space, $2D$ number of additional solution candidates is evaluated. In total, at each iteration $2D+1$ (including the current solution), the number of solution candidates is calculated in a general aspect. The algorithm begins with a random position. At the beginning of the algorithm, only one solution-candidate is calculated. At the first iteration,$2D$ number of solutions is evaluated (total$=2D+1$). At the second iteration, because the step size is fixed,$2D-1$ number of solutions is evaluated, one of which was calculated at the previous iteration (total$=4D$). This iteration is repeated until the maximum number of iteration is reached –the termination condition. Therefore, in total,$4D+(2D+1)(N-2)=N(2D+1)-2$ number of solution candidates are evaluated and compared with each other. Hence,$2(DN-1)$ number of additional calculations are made because all of the solutions are feasible. If the maximum number of iterations is 100 and, for a small-scale problem, e.g., $D = 3$, then 598 additional calculations are made. For a large-scale problem (e.g., $D = 100$) and a maximum number of iterations is 10^4, then almost 2×10^6 functions are evaluated. For the worst case of the direction line being solid, as in Fig. 1a), only N number of additional calculations is adequate. Hence, $N(2D-1)-2$ additional calculations are made. However, to increase the local search capability of the algorithm, the HJ algorithm begins with a relatively large step size; at each iteration, this step size decreases until the approximate optimal point is reached.

The convergence of the pattern search for the HJ algorithm is discussed by Torczon (1997), Dolan, Lewis, and Torczon (2003) in

detail. In the HJ algorithm, the distance between the previous solution and the current one is recorded. In Fig. 1a and b, the distance between point [A,B], [B,C], and[C,D] are the indicators of the step size (Δ). If this distance (difference between two values) is equal to zero, then this will be one of the indicators of the termination condition. If this distance is not zero, then the current step size is multiplied by a number between (0,1) ($\gamma_k\Delta_k$, $0 < \gamma_k < 1$ and $\gamma_k < \gamma_{k-1}$), where k represents the index for the iteration. Any iteration, x_N, produced by the HJ algorithm can be expressed as:

$$x_N = x_0 + \sum_{k=0}^{N-1} \Delta_k \tag{2}$$

$$\Delta_k = \gamma_1\gamma_2\gamma_3 \ldots \gamma_k\Delta_0 \tag{3}$$

where Δ_k represents the step size, Δ_0 represents the initial choice for the step size and γrepresents a number between [0,1]. Therefore:

$$\begin{aligned} x_N &= x_0 + \Delta_0 + \Delta_1 + \Delta_2 + \ldots + \Delta_{N-1} \\ &= x_0 + \Delta_0 + \gamma_1\Delta_0 + \gamma_1\gamma_2\Delta_0 + \gamma_1\gamma_2\gamma_3\Delta_0 + \ldots \\ &\quad + \gamma_1\gamma_2\gamma_3 \ldots \gamma_{N-1}\Delta_0 \\ &= x_0 + \Delta_0(1 + \gamma_1 + \gamma_1\gamma_2 + \gamma_1\gamma_2\gamma_3 + \ldots + \gamma_1\gamma_2\gamma_3 \ldots \gamma_{N-1}) \end{aligned} \tag{4}$$

Define a variable so that $\zeta_{N-1} = (1 + \gamma_1 + \gamma_1\gamma_2 + \gamma_1\gamma_2\gamma_3 + \ldots + \gamma_1\gamma_2\gamma_3 \ldots \gamma_{N-1})$; therefore, if $\zeta_{N-1} > 1$,then $x_N = x_0 + \Delta_0\zeta_{N-1}$. If the sequence x_N converges, then $\lim_{k\to\infty}(x_k - x_{k-1}) = 0$, where $x_k = x_0 + \Delta_0\zeta_{k-1}$ and $x_{k-1} = x_0 + \Delta_0\zeta_{k-2}$. Thus,

$$\begin{aligned} x_k - x_{k-1} &= (x_0 + \Delta_0\zeta_{k-1}) - (x_0 + \Delta_0\zeta_{k-2}) \\ x_k - x_{k-1} &= \Delta_0(\zeta_{k-1} - \zeta_{k-2}) \\ x_k - x_{k-1} &= \Delta_0[(1 + \gamma_1 + \gamma_1\gamma_2 + \ldots + \gamma_1 \ldots \gamma_{N-2} + \gamma_1 \ldots \gamma_{k-1}) \\ &\quad - (1 + \gamma_1 + \gamma_1\gamma_2 + \ldots + \gamma_1 \ldots \gamma_{k-2})] \\ x_k - x_{k-1} &= \Delta_0(\gamma_1\gamma_2 \ldots \gamma_{k-1}) \\ x_k - x_{k-1} &= \Delta_0 \prod_{i=1}^{k-1} \gamma_i \end{aligned} \tag{5}$$

$$\lim_{k\to\infty}\left(\Delta_0 \prod_{i=1}^{k-1}\gamma_i\right) = \Delta_0 \lim_{k\to\infty}\left(\prod_{i=1}^{k-1}\gamma_i\right) \tag{6}$$

As formulated in Eq. (6), as the variable k goes to infinity, the value of multiplier λ decreases, and the number of multipliers increases. Therefore, this multiplication becomes zero.

2.2. Proposed multiobjective Hooke–Jeeves optimization algorithm

The basic nature of the proposed algorithm depends on the solution value improvements at every iteration. The update formulation is given below:

$$\bar{x}_i[t+1] = \bar{x}_i[t] + \lambda[t] \times \overline{dr}[t] \tag{7}$$

$$\begin{bmatrix} x_{i,1}[t+1] \\ x_{i,2}[t+1] \\ \ldots \\ x_{i,d}[t+1] \end{bmatrix} = \begin{bmatrix} x_{i,1}[t] \\ x_{i,2}[t] \\ \ldots \\ x_{i,d}[t] \end{bmatrix} + \lambda[t] \begin{bmatrix} dr_1[t] \\ dr_2[t] \\ \ldots \\ dr_d[t] \end{bmatrix} \tag{8}$$

where λ represents the step size and ***dr*** represents the direction vector. The direction is the path on the decision space that improves the value of the solution by following this direction, and λ is the variable that indicates the step length on the decision space. In this research, both direction and step size are determined via systematic approaches. The four-phase design of the proposed algorithm is presented in (Algorithm 1):

Algorithm 1 Multiobjective Hooke–Jeeves optimization algorithm.

STEP 1 Beginning Phase:
Constants: Number of objectives m
Decision dimension d
Number of solutions in population (members) n
Maximum iteration t_{max}
Initialization: Initialize the population by using uniformly distributed random number generator, where the bandwidth of the random numbers lies in the search domain, initialize step size λ and $\boldsymbol{\varepsilon}$, which is a small control variable;
for $k = 1$ to n
Evaluate each member $(\bar{x}_i, i = \{1, 2, \ldots, n\})$ on all objectives $F(\bar{x})$
end for
STEP 2 Iteration Phase:
for $t = 1$ to t_{max}
for $j = 1$ to d
for $l = 1$ to n
Obtain temporary members$x_{j,l}{}^{+}[t] = x_{j,l}[t] + \lambda_j[t]$ and $x_{j,l}{}^{-}[t] = x_{j,l}[t] - \lambda_j[t]$
end
end for
STEP 3 Direction Phase
for $i = 1$ to n
Evaluate temporary members$F^{+}(\bar{x}_i^{+})$ and $F^{-}(\bar{x}_i^{-})$
if $\bar{x}_i^{+} \prec \bar{x}_i^{-}$
if $\bar{x}_i \prec \bar{x}_i^{-}$
$\bar{x}_i = \bar{x}_i^{-}$
else
$\bar{x}_i = \bar{x}_i^{+}$
end if
else if $\bar{x}_i \prec \bar{x}_i^{+}$
$\bar{x}_i = \bar{x}_i^{+}$
end if
end if
STEP 4 Update Phase
$\lambda_i[t+1] = \lambda_i[t] - \varepsilon$
end for
end for

2.2.1. Step 1 beginning phase

The algorithm begins with the initialization of the population on the decision space randomly. Uniformly distributed random numbers are generated by using a pseudorandom number generator. Each solution is evaluated on the objective function, and cost values are obtained. At this phase, the initial step length λ and control variable ε are assigned. The large initial value for the step length is assigned because it is desired that the searching begins with the boundaries of the decision space. In contrast, the control variable is assigned as small as possible. A trade-off exists between the step length and the control variable. It is desired that, at the beginning of the iteration, the algorithm searches the boundaries of decision space; as the iterations increases, more local areas of the algorithm are searched. Fig. 2 demonstrates the searching area in the decision space.

2.2.2. Step 2 iteration phase

The iteration begins at this phase of the algorithm and continues to the end of step 4. At this phase, the solutions are altered by the variable λ for each dimension as$x_{j,l}{}^{+}[t] = x_{j,l} + \lambda_j[t]$ and $x_{j,l}{}^{-}[t] = x_{j,l} - \lambda_j[t]$. The step size λ is added and subtracted from the solution candidates in each dimension. By this approach, two temporary solutions for each dimension are proposed. Fig. 3 shows the assignment of temporary solutions.

2.2.3. Step 3 direction phase

In this phase, the direction of the solution is revealed by comparing temporary solutions with the original solutions. If any temporary solution is better than the original solution, then it is replaced by a temporary solution. In other words, the solution is replaced with the nondominated solution in the set $\{x_{j,1}{}^{+}[t], x_{j,2}{}^{+}[t], \ldots, x_{j,d}{}^{+}[t], x_{j,1}{}^{-}[t], x_{j,2}{}^{-}[t], \ldots, x_{j,d}{}^{-}[t]\}$ if it exists.

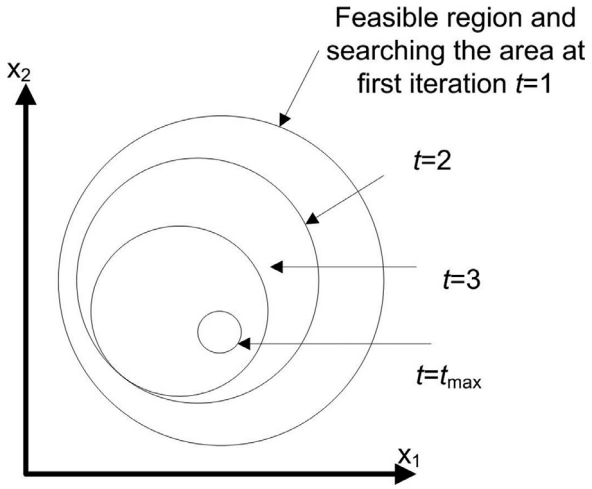

Fig. 2. Searching in the decision space in the proposed algorithm.

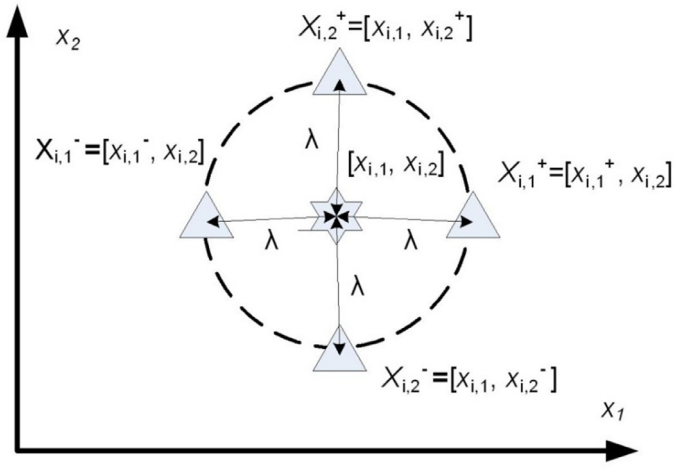

Fig. 3. Calculation of the temporary solution on the decision space for the proposed algorithm.

2.2.4. Step 4 update phase

At the end of each iteration, the variable λ is updated. In this manner, the members of the population can search a larger area of the search space. However, as the iteration increases, it is desired to focus on more local areas by decreasing the λ. This decreasing approach is shown in Fig. 4 for two-dimensional problems. Two properties for each solution/member in the population are defined for this algorithm: one is the direction, and the other is the step size. The effects of the step size and direction are presented in Figs. 2 and 4.

2.3. Differentiation and parameter selection

The proposed algorithm has a simple architecture. Every member in the decision space checks its surrounding possible solutions within a circle, which has dynamic radius/step size. If these solutions present better objective values, i.e., they dominate the existed solution, then the current member is replaced; in other words, the direction to the Pareto Front is determined. This structure is very similar to classical optimization algorithms with a numerical calculation with respect to direction and steps size. The direction of the member in MOHJ algorithm can easily be identified by examining two alternative solutions that are equal distance to each other at each dimension. This distance is equal to the step size. Therefore,

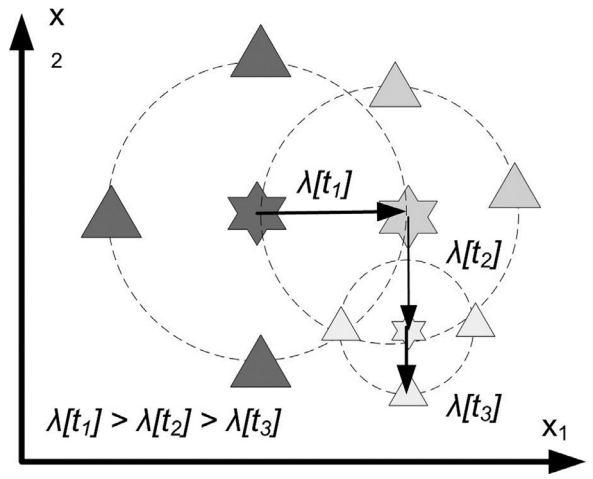

Fig. 4. Graphical demonstration of the proposed algorithm with the step size decreasing.

the determination of accurate step-size is very important. Hence, in this section, the importance of the step size and the determination of the step size will be explained by using the properties of numerical differentiation because the value of the step size is determined by using the numerical differentiation. The direction of the member in the MOHJ algorithm is based on the numerical differentiation. Two consecutive solution candidate points evaluated. Determining how far these two data points should be is the main theme of this section. Therefore, the relationship between the numerical differentiation and the step size was investigated.

The most frequently used numerical differentiation method is called the finite difference method, which was introduced by Taylor in 1715 (Reggio & Godin, 2000). By using the Taylor expansion, the numerical differentiation can be written as:

$$f'(x_0) = \frac{f(x_0+h) - f(x_0)}{h} - \frac{f^{(2)}(c)}{2} \tag{9}$$

where h represents the distance between two consecutive data samples (step size). The difference between numerical differentiation and analytical differentiation is the second part of the Eq. (9):

$$\left| f'(x_0) - f'_N(x_0) \right| = \frac{f^{(2)}(c)}{2} h \tag{10}$$

This difference is called the truncation error. Eq. (5) indicates that if the step size is selected as small as possible, then it is possible to obtain a pseudo analytical numeric differentiation equation. However, the numerical calculation can only be possible in a computer environment. Therefore, the computers are able to process to a predefined precision (i.e., it is not possible to obtain the smallest value via the computer). In IEEE 754, regarding double precision standards, the minimum number is approximately $\delta = 2^{-35} \approx 10^{-16}$. In other words, the values smaller than δ are not be able to be processed. The graphic in Fig. 5 illustrates that, for very small step size, the difference increases as the step size increases.

In Appendix A, the authors present calculations to obtain the optimum step size, which is $h_{opt} = \sqrt{\frac{2\delta f(x)}{f^{(2)}(x)}}$, where the optimum step size is approximately equal to 10^{-8}. The step size in MOHJ algorithm is not allowed to exceed that value. However, to use such a small step size in the optimization algorithm is not feasible because it would require huge amounts of function execution and time. Therefore, it is desired to initially explore the search space over as large as possible a range and then decrease the range. In this study, a Newton–Raphson-like step-size method was evalu-

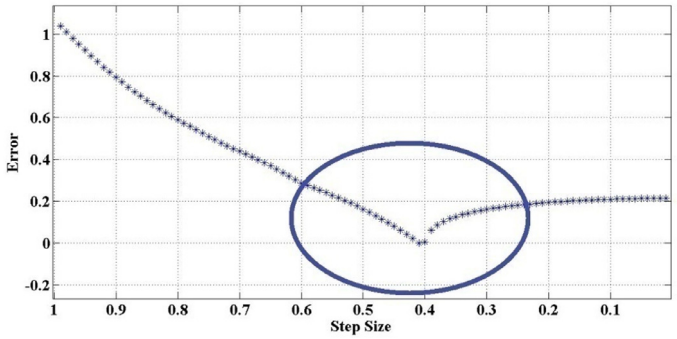

Fig. 5. Change in error with respect to the step size.

ated.

$$\lambda_i[t+1] = \lambda_i[t] - \varepsilon$$

$$\varepsilon = \max\left(\frac{\|\nabla F(\bar{x}_i)\|_2}{\|\nabla^2 F(\bar{x}_i)\|_2}, h_{opt}\right) \tag{11}$$

Eq. (11) describes the reduction of the step size. From the equation of the optimum step size, it is desired to limit the minimum step size value to a specific value of (h_{opt}) 10^{-8}. However, in the search space, some parts could not be checked. Therefore, randomization was added to the step size to obtain the stochastic property to the algorithm. Therefore, the final step size update rule is defined in Eq. (12), where *rand* is a uniformly distributed random number generator in [0,1].

$$\varepsilon = \max\left(rand\frac{\|\nabla F(\bar{x}_i)\|_2}{\|\nabla^2 F(\bar{x}_i)\|_2}, h_{opt}\right) \tag{12}$$

The proposed algorithm is a stochastic based heuristic optimization algorithm; thus, the nature of the algorithm greatly depends on the following: (i) the HJ algorithm, (ii) the domination principle and Pareto condition, and (iii)the Newton–Raphson methodology. The convergence of the proposed algorithm is evaluated similar to the HJ algorithm, where $x_N = x_{N-1} + \lambda_{N-1}$(where $\Delta = 1$). Hence, $x_N - x_{N-1} = \lambda_{N-1}$ and $\lim_{k\to\infty}(x_k - x_{k-1}) = \lim_{k\to\infty}\lambda_{k-1}$. As the k goes infinity, the step size also goes to infinity. Even from this equation, it is clear that, as the iteration goes to optimal point, more local areas are searched; hence, the step size decreases and goes to zero. However, a deeper analysis is required to obtain the relation between the initial definition of the step size and the convergence. Therefore, because $\lambda_k = \lambda_{k-1} - \varepsilon_{k-1}$, $\lambda_{k-1} = \lambda_{k-2} - \varepsilon_{k-2}$, the following is obtained:

$$\lambda_k = \lambda_0 - \sum_{i=0}^{k-1}\varepsilon_i \tag{13}$$

where λ_0 represents the initial value for the step size and ε represents the proposed update rule. From the convergence definition $\lim_{k\to\infty}(x_k - x_{k-1}) = \lim_{k\to\infty}\lambda_{k-1} = \lim_{k\to\infty}\lambda_k = \lim_{k\to\infty}(\lambda_0 - \sum_{i=0}^{k-1}\varepsilon_i)$, the number of iterations must be selected to ensure the initial value of step size approaches zero.

$$\lambda_0 = \sum_{i=0}^{k-1}\varepsilon_i \tag{14}$$

In other words,$\lambda_0 = \sum_{i=0}^{k-1}\varepsilon_i$as k approaches infinity, where ε approaches 10^{-8}. However, the summation is greatly influence on the early definitions. This equation states that, for a relatively large value of the initial step size and sufficient iterations, the algorithm converges. Because ε is not zero, then, for infinite iterations, the equation succeeded; hence, the algorithm converges. In the next section, the proposed algorithm is applied to ten multiobjective benchmark problems, and performance of the algorithm is evaluated by comparing the results with the NSGA-II algorithm.

Table 1
The basic four multiobjective problems, where n represents the decision vector size that is equals to 30 for F_1-F_3 and 10 for F_4.

Func.	Mathematical function	Search space
F_1	$f_1(x) = x_1, f_2(x) = g(x)(1 - \sqrt{\frac{f_1(x)}{g(x)}})$	$[0, 1]^n$
F_2	$f_1(x) = x_1, f_2(x) = g(x)(1 - g(x)^2)$	$[0, 1]^n$
F_3	$f_1(x) = x_1, f_2(x) = g(x)(1 - \sqrt{\frac{x_1}{g(x)}} - \frac{x_1\sin(10\pi x_1)}{g(x)})$	$[0, 1]^n$
F_4	$f_1(x) = 1 - e^{-4x_1}\sin^6(6\pi x_1), f_2(x) = 1 - (\frac{f_1(x)}{g(x)})^2$	$[0, 1]^n$
	$g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{N} x_i$	

3. Implementation

In this section, initially, the proposed algorithm was evaluated based on ten benchmark problems to evaluate the performance of the algorithm on industrial-based applications. These problems are divided into two groups with respect to the complexity of the problems. The first group of basic problems is defined in Table 1.

These functions in Table 1 are relatively easy problems. In this group of problems, the performance of the proposed method was evaluated and compared with that of the NSGA-II algorithm. The second group of functions represent relative hard problems, as defined in Table 2 (Zhang et al., 2008). The performance evaluations of the multiobjective problems are harder than performance evaluations of single-objective cases. Because, in multiobjective solutions are a set of nondominated particles, and the solution accuracy greatly depends on the distance to the Pareto Front and the distribution of the solution against the shape of the Pareto Front. Therefore, in this paper, three metrics are preferred to present more accurate information. The first metric is called the alternative Hausdorff distance metric (Schutze, Esquivel, Lara, & Coello, 2012), defined as

$$H(X,P) = \max\left(\sup_{x\in X}\inf_{p\in P} d(x,p); \sup_{x\in X}\inf_{p\in P} d(p,x)\right) \tag{15}$$

where *H(X,P)* is the Hausdorff distance metric for two data sets for solutions (X) and samples Pareto Front (P); *d(x,p)* is the Euclidean distance between solution $x \in X$and $p \in P$(in this study, the distance is simply indicated as $\bar{d}$.).In summary, this Hausdorff metric takes the maximum of two quantities: the first is the sum of the minimum distance from solution set to Pareto Front; the second is the inverse one, the sum of the distances between the Pareto Front and the solution set.

The second metric is the metric for evaluating the solution set spreading onto/near the Pareto Front. The metric is called Δ-spacing or Δ-spacing metric (While, Bradstreet, & Barone, 2012). The following equation presents the metric formulation, where d is the Euclidian distance, N is the number of the elements in the solution set, $\bar{d}'$is the average distance between all consecutive particles, $\bar{ds}$ is the distance between the first vector on the sorted Pareto Front and the final vector on sorted solution set, and $\bar{db}$ is the distance between the first vector on the sorted Pareto Front and the first vector on the sorted solution set. This metric is suitable for only two objective problems. As the dimension of objective space increases, the distribution metric is not evaluated because of the inconsistent values.

$$\Delta = \frac{\bar{ds} + \bar{db} + \sum_{i=1}^{N-1}\left|\bar{di} - \bar{d}'\right|}{\bar{ds} + \bar{db} + (N-1)\bar{d}'} \tag{16}$$

The final metric is the hypervolume (HyV) calculation. Generally, hypervolume calculation does not require the Pareto Front in-

Table 2
The seven multiobjective problems, where n represents the decision vector size, which is equal to 10 for all functions, $J_1 = \{j|j$ is an odd number and $2 \le j \le n\}$ and $J_2 = \{j|j$ is even number and $2 \le j \le n\}$.

Func.	Mathematical function
F_5	$f_1 = x_1 + (\frac{1}{2N} + \varepsilon)\lvert\sin(2N\pi x_1)\rvert + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1} h[y_j]$, $f_2 = 1 - \sqrt{x_1} + (\frac{1}{2N} + \varepsilon)\lvert\sin(2N\pi x_1)\rvert + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2} h[y_j]$
	$h(t) = 2t^2 - \cos(4\pi t) + 1$
	$y_j = x_j - \sin(6\pi x_1 + \frac{j\pi}{n})$, $j = 2, ..., n$
F_6	$f_1 = x_1 + \max\{(\frac{1}{2N} + \varepsilon), \lvert\sin(2N\pi x_1)\rvert\} + \frac{2}{\lvert J_1\rvert}(4\sum_{j\in J_1} y_j^2 - 2\prod_{j\in J_1}\cos(\frac{20y_j\pi}{\sqrt{j}}) + 2)$
	$f_2 = 1 - \sqrt{x_1} + \max\{(\frac{1}{2N} + \varepsilon), \lvert\sin(2N\pi x_1)\rvert\} + \frac{2}{\lvert J_2\rvert}(4\sum_{j\in J_2} y_j^2 - 2\prod_{j\in J_2}\cos(\frac{20y_j\pi}{\sqrt{j}}) + 2)$
	$y_j = x_j - \sin(6\pi x_1 + \frac{j\pi}{n})$, $j = 2, ..., n$
F_7	$f_1 = x_1 + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1}[x_j - \sin(6\pi x_1 + \frac{j\pi}{n})]^2$, $f_2 = 1 - \sqrt{x_1} + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2}[x_j - \sin(6\pi x_1 + \frac{j\pi}{n})]^2$,
F_8	$f_1 = x_1 + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1}[y_j]^2$ $f_2 = 1 - \sqrt{x_1} + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2}[y_j]^2$
	$y_j = \begin{cases} x_j - [0.3x_1^2\cos(24\pi x_1 + \frac{4j\pi}{n}) + 0.6x_1]\cos(6\pi x_1 + \frac{j\pi}{n}) & j \in J_1 \\ x_j - [0.3x_1^2\cos(24\pi x_1 + \frac{4j\pi}{n}) + 0.6x_1]\sin(6\pi x_1 + \frac{j\pi}{n}) & j \in J_2 \end{cases}$
F_9	$f_1 = x_1 + \frac{2}{\lvert J_1\rvert}(4\sum_{j\in J_1} y_j^2 - 2\prod_{j\in J_1}\cos(\frac{20y_j\pi}{\sqrt{j}}) + 2)$, $f_2 = 1 - \sqrt{x_1} + \frac{2}{\lvert J_2\rvert}(4\sum_{j\in J_2} y_j^2 - 2\prod_{j\in J_2}\cos(\frac{20y_j\pi}{\sqrt{j}}) + 2)$
	$y_j = x_j - x_1^{0.5(1+\frac{3(j-2)}{n-2})}$, $j = 2, ..., n$
F_{10}	$f_1 = x_1 + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1} h[y_j]$ $f_2 = 1 - \sqrt{x_1} + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2} h[y_j]$
	$h(t) = \frac{\lvert t\rvert}{1+e^{2\lvert t\rvert}}$

Table 3
The performance indicator of the NSGA-II on basic benchmark problems (the total execution time includes all independent runs).

NSGA-II		F_1	F_2	F_3	F_4
H	Min.	0.0029463	0.009222	0.0055175	0.0050612
	Max.	0.067845	0.06984	0.077734	0.068126
	Mean	0.028614	0.027154	0.047762	0.027335
	Std. Dev.	0.019022	0.013774	0.023668	0.01975
Δ	Min.	0.50882	0.43875	0.60442	0.49419
	Max.	0.81207	0.87375	0.91867	0.87187
	Mean	0.67825	0.67609	0.71838	0.6994
	Std. Dev.	0.072338	0.11203	0.074158	0.098096
HyV	Min.	1.7623	0.35558	1.1965	0.53668
	Max.	3.906	3.2185	5.0875	2.4409
	Mean	2.8862	2.0223	3.0291	1.5441
	Std. Dev.	0.56942	0.82819	1.1073	0.47035
Total Execution Time (s)		141	150	149	151

formation. However, in this paper, hypervolume calculation is used in a different manner, i.e., the difference between the hypervolume of the Pareto Front and the hypervolume (While et al., 2012) of the solution set is calculated with respect to the same reference point. Using this approach, the necessity of the reference point is eliminated. However, as the objective space dimension is larger than three, small changes at the approximated solution set could not be observed from the hypervolume metric.

First, the proposed algorithm is applied to the basic benchmark problems given in Table 1. The results were compared to a well-known and proved algorithm NSGA-II (Deb et al., 2002). The proposed algorithm was executed for a population with 20 members, for 100 iterations, and NSGA-II was executed with 20 members but for 1000 iterations. Three metric values are taken for 30 independent runs. Tables 3 and 4 present these results.

Tables3 and 4present the performance comparison between the NSGA-II and MOHJ algorithms. It can be concluded from the results that the proposed algorithm performs better than NSGA-II with respect to the convergence of the solutions to the Pareto Front. Moreover, the metric values of the proposed algorithm remain almost the same for all independent runs, indicating more stable performance. Although the convergence of the proposed algorithm is better than NSGA-II, the distribution of the solution (space averaging metric) is worse than a NSGA-II algorithm. From these results, we can conclude that for relatively easy multiobjective problems, the MOHJ algorithm presents solutions closer to the Pareto Front. Encouraged by the promising performance in these results, the authors further investigated the performance of this algorithm on harder problems, which are defined in Table 2. Table 5 presents the metric results, which corresponds to the performance difference in objective space.

From the results presented in Table 5, the same performance conclusions are made for F5 and F8 benchmark problems. However, for two benchmark problems, NSGA-II gives better results with respect to all three metric values. Another important issue related to the MOHJ algorithm is the execution time of the algorithm because two additional function evaluations are required for each dimension. For the implementations, almost the same amount of time is required for both algorithms. It is clear that, as the number of decision space dimension increases, the computation time of MOHJ algorithm is also increased. However, the nature of the code of MOHJ algorithm is more suitable for parallelization/distribution. The results show that almost 25 times faster results can be obtained by distributing the objective calculations into different computation units (Altinoz & Yilmaz, 2015).

Next, instead of two objective optimization problems, in this part, three, five, eight, and ten objective optimization benchmark problems are evaluated to evaluate the performance of the proposed method. These benchmark problems (Li, Deb, Zhang, & Suganthan, 2018) have the common formulation given below:

$$f_i = r_i\alpha_i(1 + g_i(x)) \tag{17}$$

For this purpose, two benchmark problems are selected from this set: MaOP1 and MaOP2. For MaOP1:$\alpha_1 = 1 - x_1x_2...x_{M-1}$; $\alpha_2 = 1 - x_1x_2...(1 - x_{M-1})$;...$\alpha_M = 1 - (1 - x_1)$, M is the dimension of the objective space, $r_i = 0.1 + 10(i - 1)$, N is the dimension of the decision space, and $g_i(x) = \frac{1}{N}\sum_{j=M}^{N}((x_j - 0.5)^2 + 1 - \cos(20\pi(x_j - 0.5)))$. From this formulation it is clear that the optimal point for the value of the decision space dimension is equal to 0.5, which is the midpoint of the search space. The difficulty of this problem is that it contains many local Pareto Fronts. This problem makes it easier to fall into local optimal point.

For MaOP2:$\alpha_1 = \cos(0.5x_1\pi)...\cos(0.5x_{M-1}\pi)^{p_1}$,$\alpha_2 = \cos(0.5x_1\pi)...\cos(0.5x_{M-1}\pi)^{p_2}$,...,$\alpha_M = (\sin 0.5x1\pi)^{p_M}$,$pi = \begin{cases} 4 & \mathrm{mod}(i,2) = 1 \\ 2 & otherwise \end{cases}$,$yj = \begin{cases} 0.5 & \mathrm{mod}(j,5) \neq 0 \\ \prod_{k=1}^{M-1}\sin(0.5x_k\pi) & otherwise \end{cases}$, and $g_j(x) = 200\sum_{j=M}^{N}(x_j - y_j)^2$. For this problem, the sum of all values of the even numbered objective values and the square root

Table 4
The performance indicator of the MOHJ algorithm on basic benchmark problems (the total execution time includes all independent runs).

MOHJ		F_1	F_2	F_3	F_4
H	Min.	0.0015227	0.0012929	0.0014915	0.044554
	Max.	0.0015227	0.0012929	0.0015073	0.044554
	Mean	0.0015227	0.0012929	0.0015068	0.044554
	Std. Dev.	8.8219e-19	0	2.8902e-06	7.0575e-18
Δ	Min.	0.74705	0.54207	0.76673	0.78121
	Max.	0.74705	0.54207	0.76728	0.78121
	Mean	0.74705	0.54207	0.76675	0.78121
	Std. Dev.	1.1292e-16	3.3876e-16	0.00010071	4.5168e-16
HyV	Min.	1.8421	1.9353	2.2264	0.75844
	Max.	1.8421	1.9353	2.2453	0.75844
	Mean	1.8421	1.9353	2.227	0.75844
	Std. Dev.	4.5168e-16	6.7752e-16	0.0034506	3.3876e-16
Total Execution Time (s)		98	86	104	52

Table 5
The performance indicator of the proposed method on basic benchmark problems (the total execution time includes all independent runs).

NSGA-II							
		F_5	F_6	F_7	F_8	F_9	F_{10}
H	Min.	0.014014	0.014669	0.093751	0.022108	0.18784	0.2446
	Max.	0.27207	0.077306	0.58071	0.031429	1.0041	1.3269
	Mean	0.12146	0.039713	0.20968	0.027397	0.51445	0.69735
	Std.Dev.	0.080243	0.020001	0.10066	0.00233	0.19886	0.25882
Δ	Min.	0.48773	0.53757	0.49833	0.52008	0.5318	0.50343
	Max.	0.85537	0.91132	1.0274	0.86393	1.1254	0.93808
	Mean	0.67506	0.68494	0.71483	0.67874	0.75882	0.69169
	Std.Dev.	0.11445	0.08416	0.118	0.070139	0.1456	0.1024
HyV	Min.	0.3788	1.9012	0.7364	1.6748	1.2865	2.5277
	Max.	26.9757	6.8741	71.9369	3.7994	93.0353	167.0278
	Mean	6.6084	3.8478	6.9933	2.8774	22.4421	50.8868
	Std.Dev.	6.3444	1.3245	12.8505	0.57793	21.9728	37.5032
Total Execution Time (s)		216	195	266	209	239	264
MOHJ							
		F_5	F_6	F_7	F_8	F_9	F_{10}
H	Min.	0.065136	0.014341	0.046717	0.015271	0.35839	0.50679
	Max.	0.099537	0.016344	0.36517	0.022787	0.65603	0.65466
	Mean	0.067151	0.014487	0.058907	0.015897	0.37262	0.51734
	Std.Dev.	0.0065174	0.00036866	0.058011	0.0014898	0.054288	0.027581
Δ	Min.	0.48601	0.60082	0.52419	0.65329	0.46133	0.36718
	Max.	0.55935	0.60897	0.58586	0.65516	0.50378	0.45923
	Mean	0.53856	0.60859	0.58019	0.65477	0.49755	0.37556
	Std.Dev.	0.02004	0.001469	0.013461	0.00034651	0.0088463	0.01616
HyV	Min.	1.4076	2.1346	1.7948	1.8373	0.068851	2.7256
	Max.	1.8081	2.1524	3.4038	1.8992	16.6253	11.6562
	Mean	1.7801	2.1386	1.8795	1.8448	1.8888	3.6375
	Std.Dev.	0.076464	0.0033877	0.28825	0.013884	2.7961	1.5997
Total Execution Time (s)		176	169	265	188	186	258

of odd numbered objective values must be equal to one on the Pareto Front. The nonlinear shape of the Pareto set on decision space for this problem makes solving the problem more difficult. Because these problems can be evaluated for different number of objectives, in this study, the numbers of objectives are selected as 3, 5, 8, and 10 (e.g., if the number of iterations is set at 1000, then the numbers of the populations are 300, 500, 800, and 1000, respectively). Two additional optimization algorithms, including NSGA-II, are selected to compare the proposed method: AGEII (Wagner & Neumann, 2013) and MOEAD (Zhang and Li, 2007).

Table 6 gives the numerical comparison for many objective problems. For the MaOP1 problem, the proposed method presents the best performance among all of the other optimization algorithms. The main reason is that the shape of the Pareto set on the decision space is equal to 0.5 for all dimensions. Because this value is in the midpoint of the range, and the direction to this point can be succeeded by the step size update rule systematically, the performance of the proposed method is much better than all of the algorithms. For the MaOP2 problem, the relation of the objective values on Pareto Front is tracked by the reference points, which are defined by the MOEAD algorithm. Because the algorithm is based on decomposition of the solutions into sub problems and track the solution diversity with respect to the distance to the reference points, the performance of MOEAD if the best of all the algorithms considered.

As the final discussion of this study, the performance and applicability of the proposed algorithm can be summarized with respect to the previous experiments and literature survey as follows. (a)The MOHJ algorithm presents sufficient performance against single, multiple and many objective optimization problems. Therefore, it can be applied to a large range of industrial and engineering problems. (b) The stochastic nature of the proposed method introduces randomness into the step size assignment. However, after the algorithm begins to converge, the step size reduces to the h_{opt} value. From the tables presenting the results, MOHJ gives the smallest standard deviation to reach the optimum point. This be-

Table 6
The performance indicator (Hausdorff)of the proposed method on multi- and many objective benchmark problems (the means of the results are given, and the bracket corresponds to a standard deviation of 30 independent runs).

	MOHJ		NSGA-II		MOEAD		AGE-II	
Obj	MaOP1	MaOP2	MaOP1	MaOP2	MaOP1	MaOP2	MaOP1	MaOP2
3	0.478e+0 (9.26e−4)	2.646e+0 (6.08e−6)	2.828e+0 (2.04e−1)	1.974e+1 (1.16e+0)	1.206e+0 (1.37e−1)	1.307e+0 (7.55e−1)	2.990e+0 (2.20e−1)	2.460e+1 (2.86e+0)
5	0.307e+0 (8.29e−4)	9.204e−1 (2.06e−5)	4.374e+0 (3.59e−1)	1.236e+1 (8.24e−1)	1.444e+0 (2.07e−1)	1.076e−1 (1.28e-1)	4.285e+0 (2.16e−1)	1.378e+1 (6.43e−1)
8	0.473e+0 (9.42e−2)	2.863e−1 (5.51e−7)	5.931e+0 (2.66e−1)	8.700e+0 (4.65e−1)	1.309e+0 (1.36e−1)	2.739e−2 (2.11e−2)	6.053e+0 (2.33e−1)	9.664e+0 (7.32e−1)
10	0.308e+0 (3.67e−4)	3.866e−2 (1.2e−4)	8.265e+0 (4.38e−1)	1.090e+1 (3.46e−1)	1.509e+0 (1.28e−1)	1.381e−2 (2.66e−2)	7.855e+0 (2.81e−1)	1.151e+1 (1.16e+0)

havior makes the proposed algorithm appropriate for use in industrial problems, which seek the same solution for the same initial parameters, unlike other stochastic heuristic optimization algorithms. (c) Because the algorithm focuses on the decision space, it requires the same computational cost, except for the calculation of the objective functions and domination for different number of objectives. However, as previously explained, the algorithm is greatly dependent on the size of the decision space dimension. For a bi-objective large-scale problem, the proposed algorithm requires more execution time when compared to the other heuristic algorithms. (d) Although the algorithm converges as the step size decreases, the convergence of the algorithm is dependent on the initial step size and the maximum number of iterations. Therefore, a relatively small initial step size causes early convergence and trapping in the local optima, whereas a large initial step size causes an infeasible solution set and undesirable additional computational cost. (e) The proposed algorithm can be easily distributed on the many core devices, thereby dramatically reducing the computational cost of the algorithm with respect to the preferred distribution software and hardware.

4. Conclusion

This paper presented a multiobjective Hooke–Jeeves algorithm. The performance of this proposed method was evaluated via twelve benchmark problems. The first four problems were found to be rather easier for the optimization algorithms. The performance of the proposed algorithm presented stability among all independent runs, and it presented better performance against the NSGA-II algorithm. However, instead of these basic problems, the authors applied the algorithm to harder multiobjective benchmark problems. The results also showed the performance difference between NSGA-II and MOHJ algorithm. Finally, the algorithm was applied to many objective problems with three, five, eight and ten objectives. The results showed that the performance of the proposed algorithm is greatly dependent on the shape of the Pareto set in the decision space. For a relatively easy set, the performances of many objective problems are best among the other algorithms. The proposed algorithm is not only simple to implement but also presents a relatively good performance. In conclusion, the proposed MOHJ algorithm is quite suitable for obtaining results closer to the Pareto Front. If the decision maker selects only one (or a pair of) solution(s) without considering the other possible solutions, then the MOHJ algorithm is more suitable for single-, multi- and many-objective optimization problems. The most important problem related to the MOHJ algorithm is the lack of a distribution handling mechanism and convergence dependency. Therefore, as a future study, the authors will pursue the following: (a) presenting the distributed(island model) MOHJ algorithm for many objective optimization problems to achieve a linear speedup, (b) providing a more robust version of algorithm with the aid of reference points(i.e., the R-MOHJ algorithm), (c) applying the algorithm to real-life problems on both simulation and control theory experimental sets to optimally decide the control parameters, and (d) combining the proposed algorithm with other heuristic multiobjective optimization algorithms (e.g., the genetic algorithm) and information technologies (e.g., neural networks) to apply the algorithm on machine learning applications.

Appendix-A

The differentiation equation is given in Eq. (A1), where the difference between analytical and numerical differentiation is presented in Eq. (A2); this difference is also called the truncated error.

$$f'(x_0) = \frac{f(x_0+h) - f(x_0)}{h} - \frac{f^{(2)}(c)}{2}h \tag{A1}$$

$$\left|f'(x_0) - f'_N(x_0)\right| = \frac{f^{(2)}(c)}{2}h \tag{A2}$$

Because of this difference, the numerical differential equation is defined in Eq. (A3). In addition, the difference between the two differentiations expressed in inequalities in Eqs. (A4)–(A6).

$$f'_N(x) = \frac{f(x+h) - f(x)}{h} \tag{A3}$$

$$\left|f'(x) - f'_N(x)\right| = \left|f'(x) - \frac{f(x+h)}{h} - \frac{f(x)}{h}\right| \tag{A4}$$

$$\begin{aligned}\left|f'(x) - f'_N(x)\right| &= \left|f'(x) - \frac{f(x+h)}{h} - \frac{f(x)}{h}\right| \\ &\le \frac{\delta \max(|f(x+h)|, |f(x)|)}{h}\end{aligned} \tag{A5}$$

$$\left|f'(x) - f'_N(x)\right| \le \frac{\delta \max(|f(x+h)|, |f(x)|)}{h} + \frac{f^{(2)}(c)}{2}h \tag{A6}$$

The right side of the inequality given in Eq. (A6) is the part that is desired to be zero. If this part Eq. (A7) is zero, then the difference also becomes zero.

$$H(x,h) = \frac{\delta \max(|f(x+h)|, |f(x)|)}{h} + \frac{f^{(2)}(c)}{2}h \tag{A7}$$

Therefore, change in the function *H(x,h)*concerning *h* is investigated.

$$\left.\frac{\partial H(x,h)}{\partial h}\right|_{h=h_{opt}} = 0 \tag{A8}$$

$$h_{opt} = \sqrt{\frac{2\delta \max(|f(x+h)|, |f(x)|)}{f^{(2)}(c)}} \tag{A9}$$

$$h_{opt} \approx \sqrt{\frac{2\delta f(x))}{f^{(2)}(x)}} \tag{A10}$$

Finally, the optimum is obtained in Eq. (A9), and the approximate equation is given in Eq. (A10).

References

Alatas, B., Akin, E., & Karci, A. (2008). MODENAR: Multi-objective differential evolution algorithm for mining numeric association rules. *Applied Soft Computing, 8*(1), 646–656. https://doi.org/10.1016/j.asoc.2007.05.003.

Altinoz, O. T., & Yilmaz, A. E. (2015). Parallelization of Hooke-Jeeves pattern recognition algorithm by using CUDA for GPGPU. In *2015 23nd Signal Processing and Communications Applications Conference (SIU)* (pp. 1793–1796). IEEE. https://doi.org/10.1109/SIU.2015.7130202.

Altinoz, O. T., & Altinoz, M. O. (2017). Classification of modern art movements with computational methods: Initial results. In *In* 2017 25nd signal processing and communications applications conference (SIU) (pp. 1–4).

Altinoz, O. T., Yanar, T. A., Ozguven, C., Demirdogen, G., & Yilmaz, A. E. (2017). Improved non-probabilistic roadmap method for determination of shortest nautical navigation path. In *ICEEE 2017* (pp. 261–266).

Ayala-Mató, F., Seuret-Jiménez, D., Escobedo-Alatorre, J. J., Vigil-Galán, O., & Courel, M. A. (2017). A hybrid method for solar cell parameter estimation. *Journal of Renewable and Sustainable Energy, 9*(6), 063504.

Bamdad, K., Cholette, M. E., Guan, L., & Bell, J. (2017). Ant colony algorithm for building energy optimisation problems and comparison with benchmark algorithms. *Energy and Buildings, 154*, 404–414.

Bazaraa, M. S., Sherali, H. D., & Shetty, C. M. (2006). *Nonlinear programming*. Hoboken, NJ, USA: John Wiley & Sons, Inc *https://doi.org/10.1002/0471787779*.

Brockhoff, D., & Zitzler, E. (2007). Improving hypervolume-based multiobjective evolutionary algorithms by using objective reduction methods. In *2007 IEEE congress on evolutionary computation* (pp. 2086–2093). IEEE. https://doi.org/10.1109/CEC.2007.4424730.

Coello, C. A. C., & Christiansen, A. D. (1998). Two new GA-based methods for multiobjective optimization. *Civil Engineering and Environmental Systems, 15*(3), 207–243. https://doi.org/10.1080/02630259808970240.

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. https://doi.org/10.1109/4235.996017.

Deng, T., & Li, K. (2017). Hybrid invasive weed optimization algorithm for parameter estimation of pharmacokinetic model. *International Journal of Pattern Recognition and Artificial Intelligence, 31*(3), 1759003.

Dolan, E. D., Lewis, R. M. &, & Torczon, V. J. (2003). On the local convergence of pattern search. *SIAM Journal on Optimization, 14*(2), 567–583.

Domínguez-Isidro, S., & Mezura-Montes, E. (2018). A cost-benefit local search coordination in multimeme differential evolution for constrained numerical optimization problems. *Swarm and Evolutionary Computation, 39*, 249–266.

Eklund, N. H., & Embrechts, M. J. (1999). GA-based multi-objective optimization of visible spectra for lamp design. In *Intelligent engineering systems through artificial neural networks: Vol. 9* (pp. 451–456). Retrieved from. https://www.scopus.com/inward/record.uri?eid=2-s2.0-0033338331&partnerID&40&md5=dadb4bda9a440974a59c144f0f8512e8 .

Elhossini, A., Areibi, S., & Dony, R. (2010). Strength pareto particle swarm optimization and hybrid ea-pso for multi-objective optimization. *Evolutionary Computation, 18*(1), 127–156. https://doi.org/10.1162/evco.2010.18.1.18105.

Gacôgne, L. (1997). Research of pareto set by genetic algorithm, application to multicriteria optimization of fuzzy controller. In *5th European congress on intelligent techniques and soft computing* (pp. 837–845).

Ghazzai, H., Feidi, A., Menouar, H., & Ammari, M. L. (2017). An exploratory search strategy for data routing in flying ad hoc networks. In *IEEE international symposium on personal, indoor and mobile radio communications* (pp. 1–7).

Hooke, R., & Jeeves, T. A. (1961). " Direct search" solution of numerical and statistical problems. *Journal of the ACM, 8*(2), 212–229. https://doi.org/10.1145/321062.321069.

Léon-Celi, C. F., Iglesias-Rey, P. L., Martínez-Solano, F. J., & Savic, D. (2018). Operation of multiple pumped-water sources with no storage. *Journal of Water Resources Planning and Management, 144*(9), 04018050.

Li, H., Deb, K., Zhang, Q., & Suganthan, P. N. (2018). *Challenging novel many and multi-objective bound constrained benchmark problems* (pp. 1–10) Technical Report.

Liu, D., Liu, C., Zhang, C., Xu, C., Du, Z., & &Wan, Z. (2018). Efficient hybrid algorithms to solve mixed discrete-continuous optimization problems: A comparative study. *Engineering Computations, 35*(2), 979–1002.

Merritt, F. D. (1898). Cours d'economie politique . vilfredo pareto. *Journal of Political Economy, 6*(4), 549–552. https://doi.org/10.1086/250536.

Panag, T. S., & Dhillon, J. S. (2017). Maximal coverage hybrid search algorithm for deployment in wireless sensor networks. Wireless Networks, 1-16, Article in Press.

Rahimi-Vahed, A. R., Mirghorbani, S. M., & Rabbani, M. (2007). A new particle swarm algorithm for a multi-objective mixed-model assembly line sequencing problem. *Soft Computing, 11*(10), 997–1012. https://doi.org/10.1007/s00500-007-0149-z.

Reggio, M., & Godin, D. (2000). Taylor's magic splines. *International Journal of Computer Mathematics, 75*(4), 465–480. https://doi.org/10.1080/00207160008804998.

Rosenberg, R. S. (1970). Stimulation of genetic populations with biochemical properties: I. The model. *Mathematical Biosciences, 7*(3–4), 223–257. https://doi.org/10.1016/0025-5564(70)90126-4.

Sarker, R., & Abbas, H. A. (2004). Differential evolution for solving multiobjective optimization problems. *Asia-Pacific Journal of Operational Research, 21*(2), 225–240. https://doi.org/10.1142/S0217595904000217.

Schutze, O., Esquivel, X., Lara, A., & Coello, C. A. C. (2012). Using the averaged Hausdorff Distance as a performance measure in evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation, 16*(4), 504–522. https://doi.org/10.1109/TEVC.2011.2161872.

Torczon, V. J. (1997). On the convergence of pattern search algorithms. *SIAM Journal on Optimization, 7*(1), 1–25.

Wagner, M., & Neumann, F. (2013). A fast approximation-guided evolutionary multi-objective algorithm. In *Proceedings of the 15th annual conference on genetic and evolutionary computation* (pp. 687–694).

While, L., Bradstreet, L., & Barone, L. (2012). A fast way of calculating exact hypervolumes. *IEEE Transactions on Evolutionary Computation, 16*(1), 86–95. https://doi.org/10.1109/TEVC.2010.2077298.

Yang, Z., Zhang, J., Zhou, W., & &Peng, X. (2018). Hooke-jeeves bat algorithm for systems of nonlinear equations. In *13th international conference on natural computation, fuzzy systems and knowledge discovery* (pp. 542–547).

Yang, X., & Li, J. (2010). Chaos DNA Hooke-Jeeves evolutionary algorithm and its application in river pollution control. In *International conference on mechanic automation and control engineering* (pp. 1–4).

Zhan, Y., Yang, D., Liu, Q., Zeng, C., & Wang, Y. (2018). Processing the reflectance data of rough surface for inversing the index of refraction. *Lecture Notes in Electrical Engineering, 445*, 373–382.

Zhang, H., Ma, T., Huang, G.-B., & Wang, Z. (2010). Robust global exponential synchronization of uncertain chaotic delayed neural networks via dual-stage impulsive control. *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics, 40*(3), 831–844.

Zhang, H., Zhang, J., Yang, G.-H., & Luo, Y. (2015). Leader-based optimal coordination control for the consensus problem of multiagent differential games via fuzzy adaptive dynamic programming. *IEEE Transactions on Fuzzy Systems, 23*(1), 152–163.

Zhang, H., Yang, F., Liu, X., & Zhang, Q. (2013). Stability analysis for neural networks with time-varying delay based on quadratic convex combination. *IEEE Transactions on Neural Networks and Learning Systems, 24*(4), 513–521.

Zhang, H., & Wang, Y. (2008). Stability analysis of Markovian jumping stochastic Cohen–Grossberg neural networks with mixed time delays. *IEEE Transactions on Neural Networks, 19*(2), 366–370.

Zhang, H., Cui, L., Zhang, X., & Luo, Y. (2011). Data-driven robust approximate optimal tracking control for unknown general nonlinear systems using adaptive dynamic programming method. *IEEE Transactions on Neural Networks, 22*(12), 2226–2236.

Zhang, Q., & Li, H. (2007). MOEA/D: A multi-objective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation, 11*(6), 712–731. doi:10.1109/TEVC.2007.892759.

Zhang Q., Zhou A., Zhao S., Suganthan P.N., Liu W. & Santosh T. (2008). *Multiobjective optimization Test Instances for the CEC 2009 Special Session and Competition*.

Zitzler, E., Laumanns, M., & Thiele, L. (2001). *SPEA2: improving the strength pareto evolutionary algorithm*.