Scientia Iranica D (2011) 18 (6), 1435–1442

Sharif University of Technology

Scientia Iranica

Transactions D: Computer Science & Engineering and Electrical Engineering

www.sciencedirect.com

Convex hull ranking algorithm for multi-objective evolutionary algorithms

M. Davoodi Monfared [a,*], A. Mohades [a], J. Rezaei [b]

[a] *Laboratory of Algorithms and Computational Geometry, Department of Mathematics and Computer Science, Amirkabir University of Technology, Tehran, P.O. Box 15875-4413, Iran*

[b] *Department of Technology, Strategy and Entrepreneurship, Faculty of Technology, Policy and Management, Delft University of Technology, P.O. Box 5015, The Netherlands*

Received 22 August 2010; revised 11 March 2011; accepted 2 May 2011

KEYWORDS

Convex hull;
Non-dominated solutions;
Multi-objective evolutionary algorithms;
Complexity;
Ranking procedure.

Abstract Due to many applications of multi-objective evolutionary algorithms in real world optimization problems, several studies have been done to improve these algorithms in recent years. Since most multi-objective evolutionary algorithms are based on the non-dominated principle, and their complexity depends on finding non-dominated fronts, this paper introduces a new method for ranking the solutions of an evolutionary algorithm's population. First, we investigate the relation between the convex hull and non-dominated solutions, and discuss the complexity time of the convex hull and non-dominated sorting problems. Then, we use convex hull concepts to present a new ranking procedure for multi-objective evolutionary algorithms. The proposed algorithm is very suitable for convex multi-objective optimization problems. Finally, we apply this method as an alternative ranking procedure to NSGA-II for non-dominated comparisons, and test it using some benchmark problems.

1. Introduction

Due to many applications of Multi-Objective Optimization Problems (MOOP) in industry and other real world problems (see for example [1–3]), solving MOOPs has become an attractive research topic in recent years. Since many of these problems belong to the NP-Hard class of problems, salient studies have been done to solve MOOPs by heuristic algorithms (see for example [4–10]). The Genetic Algorithm (GA) is one of the most popular approaches in this context [11]. The concept of GA was first developed by Holland [12], and has been since improved [13–15] with respect to the convergence rate and quality of final solutions. Most recent improvements are on Multi-Objective Optimization (MOO) algorithms. There are two goals in a MOOP: first, finding a set of solutions as close as possible to the Pareto-optimal front (trade-off solutions), and second finding a set of solutions as diverse as possible in the obtained non-dominated front.

Most evolutionary studies focus on these two goals. NSGA-II and SPEA2 are among the most powerful evolutionary methods, which are evolved versions of NSGA and SPEA, respectively [16,17]. Although by using non-dominated principles, the power of Multi-Objective Evolutionary Algorithms (MOEAs) is enhanced to reach the main goals of MOO, they have still some weaknesses and there are several test problems that these algorithms are not strong enough to solve [11].

Since most computational geometric algorithms are exact and optimal in the complexity time term, we introduce and discuss the relation between convex hull and non-dominated solutions. Geometric objects, like convex hull and onion peeling, have not appeared in MOEA literature up to now, and this work introduces a new useful connection between these two fields. We also use the convex hull to present a new ranking algorithm to cluster the solutions of a genetic population in geometric stand points. This new algorithm, called the Convex Hull Ranking Algorithm (CHRA), uses geometric objects, like convex hull and onion layers, and is very suitable for convex MOOPs. CHRA is a powerful approach to improve the convergence rate of solutions to the Pareto-optimal front in

* Corresponding author.
E-mail address: mdmonfared@aut.ac.ir (M. Davoodi Monfared).

1026-3098 © 2012 Sharif University of Technology. Production and hosting by Elsevier B.V. Open access under CC BY-NC-ND license.
Peer review under responsibility of Sharif University of Technology.

doi:10.1016/j.scient.2011.08.017

Production and hosting by Elsevier

convex MOOP. It is a geometric ranking method that is easy to imagine and can be applied instead of the non-domination principle. Another advantage of CHRA is that it can reach both mentioned goals of MOO, simultaneously; although it does not use any niching or crowding distance that are applied in other MOEAs [11]. Finding a diverse set of Pareto-optimal solutions in most MOEAs is another bottleneck step of algorithms in terms of complexity, i.e. they need to apply some crowding procedures or niching operators, whose complexity time is $O(MN^2)$ or $O(MN \log N)$, where M is the number of objectives and N is the size of the GA population. CHRA is a general ranking method that can be used in any MOEA. In addition, we present a compatible version of CHRA to solve the constrained MOOPs and propose some approaches that can be used with CHRA to satisfy the constraints.

The rest of this paper is organized as follows. Section 2 gives an overview of two geometric objects that are used in CHRA. Section 3 describes some challenging problems of MOEAs and introduces NSGA-II briefly. Section 4 discusses the relation between convex hull and non-dominated solutions, and proposes CHRA. Section 5 contains simulation results obtained from the algorithm, while Section 6 discusses the properties of CHRA. Concluding remarks and future work are given in the final section.

2. Geometric preliminary

In this section, we make an overview of two geometric objects, convex hull and onion peeling used in our algorithm.

2.1. Convex hull

Convex hull is a well-known geometric object utilized in many systems, such as shape analysis, robotics (collision avoidance), geographical information system, location, and assessment of roundness error [18]. In general, convex hull is defined on a set of points in d-dimensional ($d > 1$) space. In the following, there are two definitions of convex hull on the plane.

Definition (*Convex Hull*)**.**

1. The smallest convex *polygon* that encloses all points of a set.
2. The intersection of all *halfplanes* that contains all points of a set.

To define the convex hull in high dimensions, it is sufficient to replace *polygon* and *halfplane* with *polyhedral* and *halfspace* in the above definitions, respectively.

Let $P = \{p_1, p_2, \ldots, p_n\}$ be a set of n points in the plane. The convex hull of P, denoted by $\mathrm{CH}(P)$ is a sequence of points like $\mathrm{CH}(P) = (p_{i_1}, p_{i_2}, \ldots, p_{i_m})$ where $m \leq n$ and $p_{i_j} \in P$ for $j = 1, 2, \ldots, m$. So, that covers all points of P. Figure 1 shows a set of points and its convex hull.

Several algorithms have been suggested to calculate a convex hull in the plane, some of which can be extended to higher dimensions. Simple algorithms for computing the convex hull in the plane test all pairs of points, and find extreme points or edges (point or edge that lies on the convex hull). These algorithms run in $O(n^4)$ and $O(n^3)$ time, respectively. Also, there are several algorithms that work in $O(n^2)$ time, like the gift-wrapping algorithm and the incremental approach [18]. Reducing the sorting problem to the convex hull problem, it is proved that all algorithms which compute the convex hull of n points in the plane belong to $\Omega(n \log n)$. In fact, this time is the lowest bound for convex hull algorithms. Fortunately, there is the optimal Graham algorithm that works in $O(n \log n)$ and the Quick Hull algorithm that runs better in average in $O(n \log n)$, too [19]. Because of simple implementation and optimal time, the Graham scan is a popular algorithm for computing a convex hull. In the following, a pseudo code version of the Graham algorithm is presented:

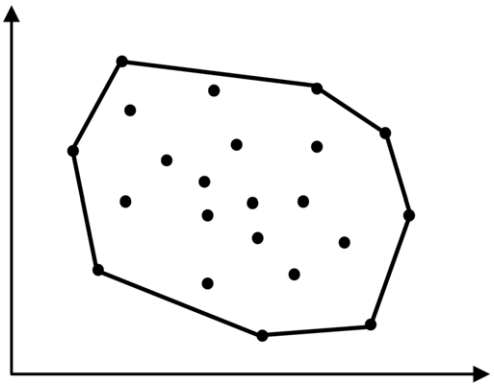

Figure 1: An example of convex hull in the plane.

Algorithm: Graham scan:

Input: Set of points $P = \{p_1, p_2, \ldots, p_n\}$.
Output: Convex hull of P.
Step 1: Find rightmost lowest point; label it p_0.
Step 2: Sort all other points angularly about p_0.
(in case of tie, delete the point closer to p_0).
Step 3: Create stack $S = (p_1, p_0) = (p_t, p_{t-1})$. Set t as its top index, and set variable i to 2.
Step 4: While $i < n$, do following steps:
If p_i is strictly left of $p_{t-1}p_t$, then
Push(p_i, S) and set $i = i + 1$
Else *Pop*(S).

The algorithm uses a stack S and pushes the rightmost lowest point in it. It finds one extreme point in each iteration and works in a counter clockwise form. Detailed and exact implementation of this algorithm can be found in [18].

Chan [20] presented an elegant approach for output-sensitive construction of a convex hull using ray shooting, which achieves optimal $\theta(n \log h)$ time for two and three dimensions, where h denotes the number of points of the convex hull. Also, in [21], an $O(n)$ expected time algorithm was proposed to compute the convex hull in the plane.

2.2. Onion peeling

One other useful geometric object is onion peeling (in fact, it is a process that makes the object). It is an extended convex hull object utilized in image processing and finger-printing, etc. [18]. The onion peeling of a given set, P, is a set of convex layers, defined as follows: the first layer is the $\mathrm{CH}(P)$, and the second is the convex hull of a set of P, minus the points on the first layer. The remaining layers are computed similarly. The process continues until the empty set is reached. This iterative process is called onion peeling, which leads to the onion layers. Figure 2 shows the onion layers and their number.

The number of layers is called the depth of onion, which is equal to the number of iterations of the convex hull computation process. In Figure 2, the onion peeling has been

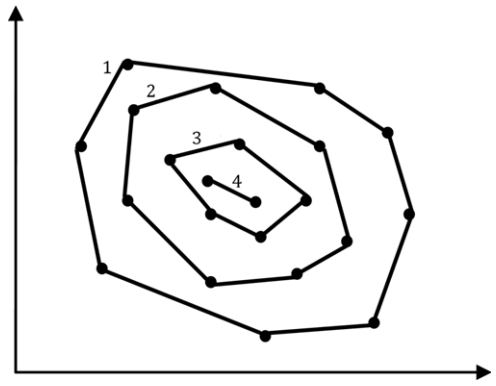

Figure 2: Onion peeling and layer numbers, the depth is four.

computed through four iterations, therefore, the onion depth equals four. Chazelle [22] proved that the onion layers could be computed in $O(n \log n)$ time using the same iterative process.

2.3. Convex hull in higher dimensions

Like two-dimensional space, there are some optimal algorithms that compute a convex hull in three-dimensional space. These algorithms run in $O(n \log n)$ expected time [19], where n is the number of points. Also in [18], an incremental approach algorithm was presented that runs in $O(n^2)$ time and computes the convex hull and also a divide and conquer algorithm in $O(n \log n)$ time. Since the lowest bound for computing a convex hull in d-dimensional space belongs to $\Omega(n \log n + n^{\lfloor d/2 \rfloor})$, computing a convex hull at higher dimensions must take extensive time [19]. Chan [20] designed an output-sensitive algorithm for a convex hull in d dimensions, which runs in $O(n \log^{O(1)} h + h^{\lfloor d/2 \rfloor})$ time, where h denotes the number of points of the convex hull.

Most real MOOPs have two or three distinctive objectives [23]. Fortunately, as mentioned above, there are optimal geometric algorithms in these cases to compute the convex hull and onion layers. In Section 4, we use these objects to propose an optimal ranking approach for two- or three-objective space. Although these algorithms can be applied at higher dimensions ($d \geq 4$), they are not optimal in terms of time.

3. Multi-objective evolutionary algorithms

A multi-objective optimization problem can be written as follows [11]:

$$\text{Minimize/Maximize } f_m(X), \quad m = 1, 2, \ldots, M.$$

Subjectto :

$$\begin{aligned} &g_j(X) \geq 0 && j = 1, 2, \ldots, J, \\ &h_k(X) = 0 && k = 1, 2, \ldots, K, \\ &X = (x_1, x_2, \ldots, x_n)^T && x_i^L \leq x_i \leq x_i^U, \text{ for } i = 1, 2, \ldots, n \end{aligned}$$

where X is a set of n decision variables, $f_m(X)$ is the mth objective function, and g_j and h_k show *soft* and *hard* constraints, respectively. Also, x_i^L and x_i^U are lower and upper bounds for variable x_i, respectively. Usually, the objectives in real world problems conflict with each other. This means that each solution can be improved toward one single objective, but that in general, some or all other objectives are sacrificed. Therefore, there is no single optimal solution for MOOPs. A traditional weighted linear combination approach [24] can be used to solve these problems, but assigning suitable weights to the objectives is a difficult task in an arbitrary problem and makes them subjective to the user. In most MOOPs, there is no proportion between the search space and the objective space, especially in problems that have a non-continuous or non-uniform diversity search space. In order to overcome this problem, MOEAs have been proposed. In fact, the goal of MOEAs is to find all Pareto-optimal solutions in a single run (in most cases, the set of Pareto-optimal solutions is infinite). In general, MOEAs use the non-dominated principle and satisfy two orthogonal objectives simultaneously:

1. Finding a set of Pareto-optimal solutions,
2. Finding a set of diverse solutions in objective search space.

Many algorithms based on the aforementioned principle are described in [11]. Because of the large number of these algorithms, they have been clustered and compared in several studies [25,26]. Figure 3 shows the performance of MOEAs on a given search space. An ideal MOEA tries to close the Pareto-optimal front by preserving diversity among its population. It needs two types of search for satisfying both goals: *along* and *lateral*. These two searches are orthogonal to each other [11] (see Figure 4(a)).

3.1. Elitist non-dominated sorting genetic algorithm

NSGA-II has been proposed by Deb et al. [27]. This algorithm uses a fixed-size population and divides the population to several subpopulations that are different on the preferred non-domination levels. In fact, solution q is dominated by solution p if, and only if, p is better than q in all objectives, or p is better than q, at least, in one objective, and is the same as q in other objectives. This rule is called the non-dominated principle. The property of the ith subpopulation (denoted by F_i) is that all its solutions are dominated by all solutions of each subpopulation placed under it ($F_j j < i$), and dominates all solutions that belong to subpopulations that are placed above it ($F_j j > i$).

Also, NSGA-II uses a global elitist strategy and tries to satisfy the second goal of MOEAs simultaneously by convergence to the Pareto-optimal solutions. If P is the parent population, Q is the child population, which is generated using crossover and mutation operators, and N is the size of the populations (parent and offspring), the NSGA-II is outlined as follows [11]:

NSGA-II algorithm:

Step1: Combine parent, P, and offspring, Q, populations and create $R_t = P_t \cup Q_t$. Perform a non-dominated sorting to R_t and identify different fronts: $F_i, i = 1, 2, \ldots$, etc.
Step2: Set new population $P_{t+1} = \varnothing$ and $i = 1$. Until $|P_{t+1}| + |F_i| < N$ perform $P_{t+1} = P_{t+1} \cup F_i$ and $i = i + 1$.
Step3: For reminder capacity in P_{t+1}, perform the crowding operator and fill it by some of the best solutions in F_i.
Step4: Create offspring population Q_{t+1} from P_{t+1} by using some of crossover and mutation operators.

The first step of the algorithm divides the combined parent and child population into several subpopulations, F_i, $i = 1, 2, \ldots,$ etc. In fact, this step is a ranking procedure. The complexity of this step is $O(MN^2)$, where N is the size of the population and M is the number of objectives. Step 3 selects some solutions that are placed in a non-crowded search area among other solutions and requires $O(MN \log N)$ computations. The crowding distance operator finds the minimum distance between a solution and other solutions in the objective space. A solution with the furthest position, in respect to other solutions, has good diversity. Other steps run in $O(N)$ time, so the total

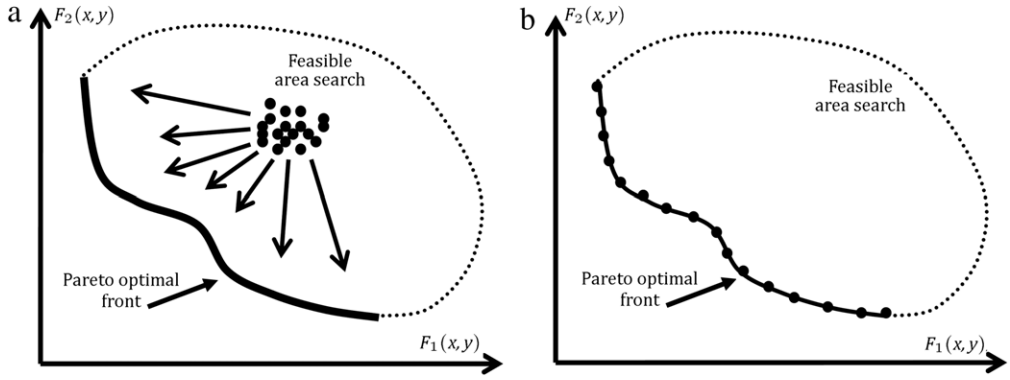

Figure 3: (a) A hypothetical search space with Pareto-optimal front. (b) An ideal non-dominated solution set.

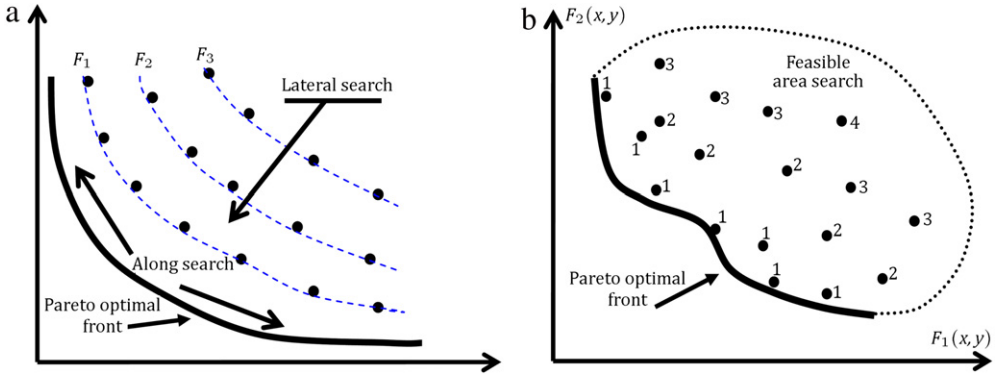

Figure 4: (a) Schematic views of the goals of MOEAs. (b) Solution ranking by non-dominated principle.

complexity of NSGA-II is $O(MN^2)$ time, which is similar to other MOEAs.

Step 1 is the bottleneck time complexity of NSGA-II (like other MOEAs). It uses the non-dominated principle and creates several non-dominated subpopulations. These subpopulations are constructed based on the degree of their closeness to the Pareto-optimal front (or user preferred levels). Figure 4(a) shows these facts. Figure 4(b) displays a hypothetical two-objective problem. In this problem, both objectives ($F_1(x, y)$ and $F_2(x, y)$) are minimization. The numbers that are close to each solution show the rank of the solution based on non-dominated sorting.

Deb et al. [28] also presented a powerful version of NSGA-II (they called it Controlled NSGA-II), which tries to reach the second goal (diversity) in a regular manner. The abovementioned complexity time of NSGA-II (and most MOEAs) is in the ranking procedure (Step 1 of the algorithm) and the $O(MN^2)$ computations. Jensen [29], using an elegant sweep line approach, improved this time to $O(N \log N)$ in the plane. He also proposed a divide and conquer algorithm in general cases, and found all non-dominated fronts in $O(N \log^{M-1} N)$ time.

4. Convex hull and non-dominated solutions

In this section, we describe the main idea of this paper and the relation between the convex hull and the first front of non-dominated solutions, as well as the relation between onion peeling and non-dominated sorting. This relation is discussed in two-dimensional space, although it is generally true for higher dimensions.

4.1. The relation between convex hull and non-dominated solutions

As mentioned in the previous section, Chazelle [22] computes the onion layers of a set of n points in $O(n \log n)$ time. Since having two objectives, there are only four case optimizations {(Min, Min), (Min, Max), (Max, Min), (Max, Max)} (see Figure 5), the extreme points on the convex hull can be divided into four clusters. The first cluster (all points between a and b in Figure 5) contains *all the best solutions* of the convex hull or the solutions that are the best with respect to both objectives (assume a minimization problem). The rank of these solutions can be set by the onion layer number. The second and third clusters contain *the middle solutions*, which are good in only one objective (all points between b and c, and between d and a in Figure 5). The solutions of the final cluster are *the worst solutions*, with respect to both objectives (all points between c and d in Figure 5). One fast algorithm for computing the convex hull is the Quick Hull [19]. This algorithm can be efficiently used for finding each quarter part of a convex hull. For example, we can use the Quick Hull only for finding a quarter of the convex hull between a and b in Figure 5. By using the middle point as a pivot point in the Quick Hull, the run time of the algorithm is guaranteed in $O(n \log n)$ time. Also, after computing the convex hull by a geometric approach, four clusters can be determined in $O(h)$ time, where h is the number of points (solutions) of the convex hull. This approach is a simple 'walking on the edges' of the convex hull and clustered points, with regard to the left or right turn relative to objective directions [18]. Note that it is possible for some of these clusters to be empty (when two of four extreme points are equal).

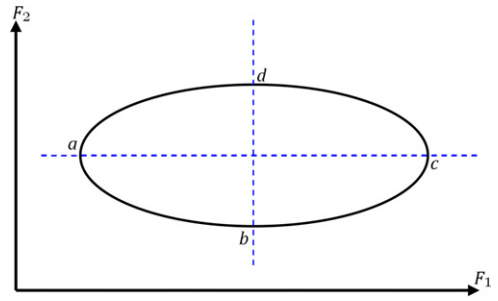

Figure 5: Four quarter part of the convex hull.

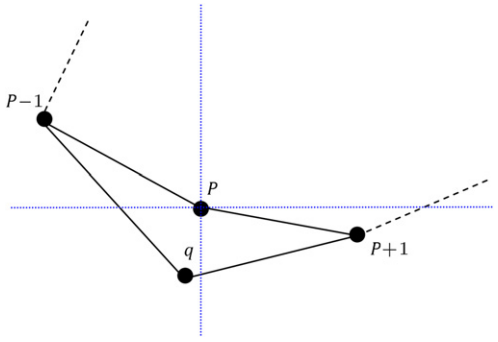

Figure 6: Each point on the convex hull is a non-dominated solution, but it is not true in the opposite direction.

It is clear that each point on the quarter part of the convex hull is a non-dominated solution [30]. However, it is not always true in the opposite direction, because it is possible that a point is a non-dominated solution, but is not in the convex hull (see Figure 6). Therefore, we can compute the first quarter of the convex hull as the first non-dominated front of the solutions to a genetic population, assign the maximum fitness to the solutions, then remove them from the population and repeat this process for the second, the third, etc. non-dominated fronts of the populations. This is the main idea in the relation between convex hulls and non-dominated solutions. By repeating this process, we also are able to assign a proper rank to each solution in the population, with respect to its degree of closeness to the Pareto-optimal front. The following theorem demonstrates that the convex hull in the plane can be obtained from non-dominated solutions and vice versa.

Theorem 1. *Convex hull and non-dominated problems are reducible to each other in two dimensions.*

Proof. First, suppose F_1 be the non-dominated solution of a genetic population. By using the Jensen algorithm in two dimensions [29], we can find this front in $\Omega(n \log n)$ computations. The algorithm reports the solutions of the first non-dominated front (also other fronts) by decreasing the order, with respect to the second objective. Therefore, the output of the algorithm is a monotone chain that starts from the leftmost points and ends at the bottommost point. Since the convex hull of a polygon (especially, in our case, a monotone chain) can be reached in linear time [18], we can compute a quarter of the convex hull from the non-dominated solutions. For the other three quarters of the convex hull, it needs to obtain non-dominated solutions for other cases of MOOPs ((Min, Max), (Max, Min), (Max, Max)).

To prove the other side of Theorem 1, let us assume that there exists a convex hull of a population of solutions, and we wish to find non-dominated solutions (all points lie on the left-bottom quarter of the convex hull or on the smallest rectangle that includes two extreme points (the best solutions, with respect to the first and the second objectives)). In this case, we use a similar sweep line approach, which starts from the left point on the convex hull and ends at the bottom point of the convex hull, and insert a point into the non-dominated solution, if the monotonic property of the created chain holds [29]. □

It is proved that the convex hull of n points in the plane belongs to $\Omega(n \log n)$ time, so finding all non-dominated solutions belongs to $\Omega(n \log n)$, too.

Each point on the boundary of the convex hull is a non-dominated solution, but it is not true for the opposite direction, which is why the number of convex hull iterations for ranking the solutions is equal to, or greater than, the number of fronts obtained by using a non-dominated principle. In the following, we present a ranking procedure based on the convex hull to rank the population of a MOEA. This ranking algorithm can be used for any multi-objective evolutionary algorithm. It is especially very suitable and fast for convex MOOP.

The convex hull and the non-dominated front are the same at higher dimensions. So, the aforementioned relations are held between them at higher dimensions, even though computing a convex hull at higher dimensions is more time consuming. That is, it is done in $O(n \log^{O(1)} h + h^{\lfloor d/2 \rfloor})$ time, where d is the size of the space; however, the first non-dominated front of solutions can be obtained in $O(n \log^{d-2} n)$ time [31]. Therefore, a challenging problem is in using non-dominated solutions for obtaining an approximate convex hull or onion peeling at higher dimensions.

The other important advantage of the convex hull ranking idea is that it takes into account both goals of MOEA: convergence to the Pareto-optimal front and diversity, simultaneously. This means that the algorithm, without using any niche approach or crowding distance, is able to find a good set of diverse solutions. In order to reach this goal, it is sufficient to use the Quick Hull algorithm for computing the convex hull with the farthest point as a pivot point. In this case, a sequence of obtained points is accepted for the last front of the non-dominated solutions.

4.2. Convex hull ranking algorithm

In this subsection, considering Theorem 1 and the relation between convex hull and non-dominated sorting, we propose a new ranking algorithm to the genetic population of a MOEA. The algorithm assigns a rank to each solution as follows. First, the non-dominated front of the solutions (the best solutions of the population) get a rank equal to one and at each iteration, the rank of the next solution is increased. So, a solution with lower rank is better than one with a higher rank.

The Convex Hull Ranking Algorithm (CHRA) in the plane first finds two especial solutions: the bottommost and the leftmost (the best solution, with regard to each objective). It then computes a quarter of the convex hull, which is between the two points, and assigns an equal rank to all solutions placed in that quarter. In the higher dimensions (for example for M objectives), it needs to find M especial solutions, each of which is the best solution with regard to one objective. It then computes parts of the convex hull that ends in these points.

The input of CHRA are members of a genetic population (denoted by *Pop* in the following code), while the output is the ranking of all members, with regard to their degree of closeness

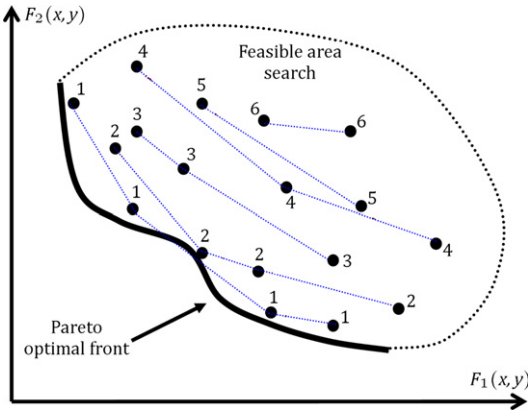

Figure 7: Solution ranking by CHRA.

to the Pareto-optimal front. In the following, the outline of CHRA is presented:

Convex hull ranking algorithm:

Input: Set of solutions, *Pop*. Output: Assigning a proper ranking number to each solution.

Step1: Set rank variable r to one.
Step2: Find the minimum solution in F_1 and F_2, call them a and b.
Step3: Compute the convex hull between a and b.
Step4: Assign the rank r to all solutions lying on the convex hull and remove them from *Pop* temporarily.
Step5: If *Pop* is empty, the algorithm is completed, otherwise set $r = r + 1$ and go to Step 2.

It is likely that some of the extreme points will be equal. For example, if the above extreme points, a and b, are equal, we can set their rank to r, remove them from the population and continue the algorithm from Step 5.

The complexity of the algorithm is similar to the onion peeling algorithm, and it runs in $O(N \log N)$ time for two and three objectives, where N is the size of the population. CHRA allocates a suitable rank to each solution without doing a non-dominated comparison. In Figure 7, we rank the solutions by CHRA. These solutions are the same as those illustrated in Figure 4(b).

5. Simulation results

In this section, we test CHAR by two test problems; a convex and a non-convex Pareto-optimal front. We report the number of generations and genetic parameters, and show the population in each simulation. We apply NSGA-II with CHRA to two test problems in which we use population size 20, crossover probability 0.9 and the binary coding of decision variables.

5.1. First test (convex Pareto-optimal front)

We use the Min-Ex test problem suggested in [11], which has been used to test many MOEAs. The model of Min-Ex is as follows:

$$\text{Min-Ex}: \begin{cases} \text{Minimize} & f_1(x_1, x_2) = x_1, \\ \text{Minimize} & f_2(x_1, x_2) = \dfrac{1 + x_2}{x_1}, \\ \text{s.t.:} & 0.1 \le x_1 \le 1,\ 0 \le x_2 \le 5. \end{cases}$$

We use a bit-wise mutation by probability $(1/\textit{length of string})$ and the variables are coded with 24 and 26 bits. Figure 8 shows the results.

5.2. Second test (non-convex Pareto-optimal front)

For the second test, we use the Fonseca and Fleming problem [11], as used in many studies:

FON :

$$\begin{cases} \text{Minimize} & f_1(x) = 1 - \exp\left(-\sum_{i=1}^{n}\left(x_i - \dfrac{1}{\sqrt{n}}\right)^2\right), \\ \text{Minimize} & f_2(x) = 1 - \exp\left(-\sum_{i=1}^{n}\left(x_i + \dfrac{1}{\sqrt{n}}\right)^2\right), \\ \text{s.t.} & -4 \le x_i \le 4, \ \text{for } i = 1, 2, 3, \ldots, n. \end{cases}$$

The Pareto-optimal solution of this problem is $x_1^* \in [-1/\sqrt{n}, 1/\sqrt{n}]$ for $i = 1, 2, \ldots, n$.

The Pareto-optimal front is a non-convex set in the FON problem. To solve this problem, we use the same parameters and mutation operator. Figure 9(a) and (b) show the population after 80 and 200 generations, respectively.

6. Discussion

In this section, we analyze the proposed ranking algorithm and discuss its advantages and disadvantages.

Since each solution on the convex hull is a non-dominated solution, the number of preferred levels in CHRA is more than that of the non-dominated comparison approach, so the nearest solutions to the Pareto-optimal front are emphasized by a selection operator, and the convergence rate increases.

Since both goals of MOEAs (convergence to the Pareto-optimal front and diversity of the final solutions) are orthogonal to each other, most MOEAs use several techniques, like niche operators and crowding distance [11]. These techniques can be used in CHRA, as well. CHRA is able to satisfy both goals, simultaneously. For example, in NSGA-II, after constructing the non-domination fronts, the next population is filled using the ordered fronts. If the population has an empty place, the front members are inserted into the population, otherwise for the last front in which there is not sufficient room in the next generation for all its members, the following approach is used.

First, a crowding distance (normalized distance to the closest neighbors in the front of the objective space) is assigned to all members of the last front, and then some of the best crowding distance solutions are selected. By using the Quick Hull algorithm for computing the convex hull, CHRA can select the best crowding distance solutions without the direct computing of any metric distances. To achieve this goal, it is sufficient that after finding the two extreme points (denoted by a and b in Figure 10), the farthest solution from them is selected as a pivot point in the Quick Hull algorithm [18]. This process continues, while the next generation of the population is filled.

The proposed algorithm in this paper can be easily generalized to handle the constrained MOOPs. For example, one technique that can be used to have a compatible CHRA in constrained optimizations is the penalty function [32,33]. Moreover, we can modify Step 4 of CHRA. To end this, before setting a rank to a solution, we can check whether the solution is feasible or not; if so, set r to the rank of solution, otherwise, set r and increase a proper value to the rank of the solution as its penalty value. The increased value must be proportionate to the opposite of r. That is, if p and q are two infeasible solutions and the rank of p (denoted by r_p) is greater than that of q (denoted by r_q), p is better than q.

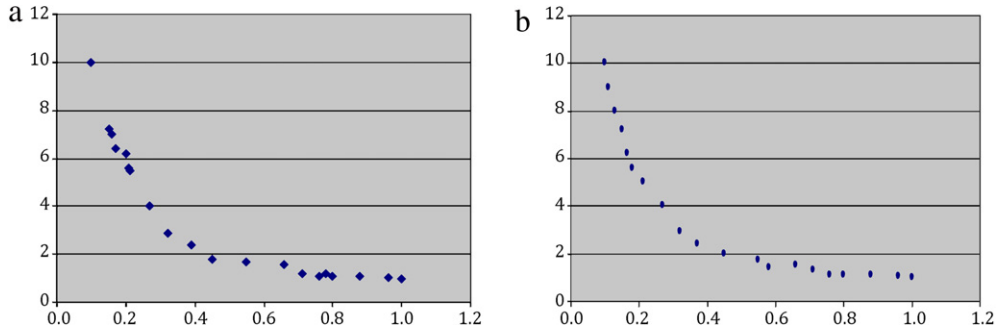

Figure 8: Results of Min-Ex problem; population after 40 and 200 generations with mutation: (a) and (b), respectively.

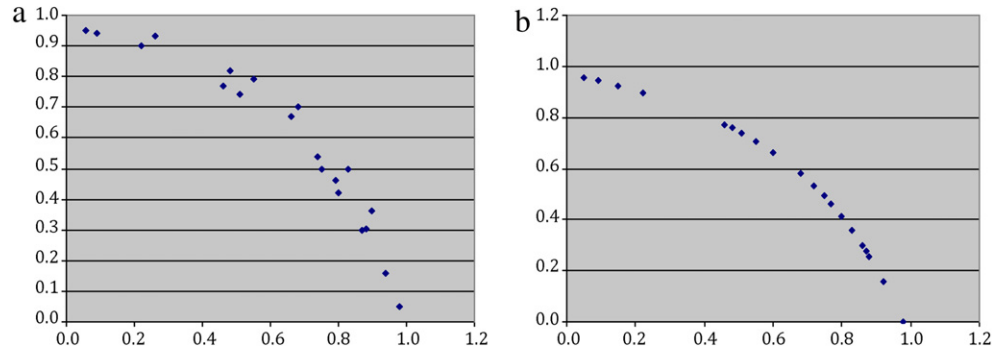

Figure 9: Results of FON problem; population after 80 generations (a) and 200 generations (b).

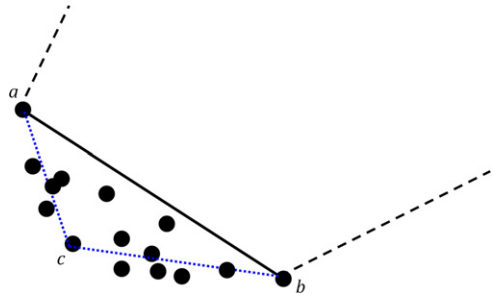

Figure 10: Using the farthest point in Quick Hull algorithm, the diversity can be guaranteed by CHRA.

Another simple approach to having a compatible CHRA in constrained optimization models is to use the tournament selection operator with the following rule. Let p and q be two solutions in the population. p is better than q if, and only if, the following three situations arise:

1. p is feasible and q is infeasible.
2. p and q are feasible and $r_p < r_q$.
3. p and q are infeasible and $r_p > r_q$.

This rule first emphasizes the feasible solutions and then the solutions closed to the Pareto-optimal front.

7. Conclusion

This paper investigates the relation between convex hull and non-dominated solutions and proposes a new simple ranking algorithm (convex hull ranking algorithm-CHRA) for ranking the solutions of the population of an evolutionary algorithm. This algorithm uses simple geometric approaches to assign an appropriate rank to the solutions of a population. The proposed ranking algorithm (CHRA) is used as an alternative ranking procedure of NSGA-II and is tested by some test problems. The results show that the algorithm is able to achieve the two main goals of multi-objective evolutionary algorithms. A challenging problem is to propose an approximate polynomial convex hull algorithm for higher dimensions using non-dominated ranking procedures.

References

[1] Rezaei, J. and Davoodi, M. "Multi-objective models for lot-sizing with supplier selection", *International Journal of Production Economics*, 130(1), pp. 77–86 (2011).
[2] Moslehi, G. and Mahnam, M. "A Pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search", *International Journal of Production Economics*, 129(1), pp. 14–22 (2011).
[3] Davoodi, M., Mohades, A. and Rezaei, J. "Solving the constrained p-center problem using heuristic algorithms", *Applied Soft Computing*, 11(4), pp. 3321–3328 (2011).
[4] Li, Z., Rudolph, G. and Li, K. "Convergence performance comparison of quantum-inspired multi-objective evolutionary algorithms", *Computer and Mathematics with Applications*, 57(11–12), pp. 1843–1854 (2009).
[5] Dehuri, S. and Mall, R. "Predictive and comprehensible rule discovery using a multi-objective genetic algorithm", *Knowledge-Based Systems*, 19(6), pp. 413–421 (2006).
[6] Haupt, S.E. "A quadratic empirical model formulation for dynamical systems using a genetic algorithm", *Computer and Mathematics with Applications*, 51(3–4), pp. 431–440 (2006).
[7] Fukuyama, Y. "Comparative studies of particle swarm optimization techniques for reactive power allocation planning in power systems", *Electrical Engineering in Japan*, 153(1), pp. 690–696 (2005).
[8] Zeng, X. and Liu, Z. "A learning automata based algorithm for optimization of continuous complex functions", *Information Sciences*, 174(3–4), pp. 165–175 (2005).
[9] Glover, F. "Tabu search—part I", *ORSA Journal on Computing*, 1(3), pp. 190–206 (1989).
[10] Davoodi, M., Mohades, A. and Rezaei, J., et al. "A genetic algorithm for the constrained coverage problem", in: J. Mehnen (Ed.), *Applications of Soft Computing, AISC*, vol. 58, pp. 347–356, Springer-Verlag, Berlin, Heidelberg (2009).

[11] Deb, K., *Multi-Objective Optimization Using Evolutionary Algorithms*, Wiley (2001).
[12] Holland, J., *Adaption in Natural and Artificial System*, University of Michigan Press, Ann Arbor, Michigan, USA (1975).
[13] Goldberg, D.E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley (1989).
[14] Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, In *AI Series*, Springer Verlag, New York (1994).
[15] Augusto, O.B., Rabeau, S., Depince, Ph. and Bennis, F. "Multi-objective genetic algorithms: a way to improve the convergence rate", *Engineering Applications of Artificial Intelligence*, 19(5), pp. 501–510 (2006).
[16] Srinivas, N. and Deb, K. "Multi-objective optimization using non-dominated sorting in genetic algorithms", *Journal of Evolutionary Computation*, 2(3), pp. 221–248 (1994).
[17] Zitzler, E., Laumanns, M. and Thiele, L., *SPEA2: improving the strength Pareto evolutionary algorithm for multi-objective optimization*, In *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, vol. 103, pp. 95–100, (2001).
[18] O'Rourke, J., *Computational Geometry in C*, 2nd ed., Cambridge University Press, New York (1998).
[19] de Berg, M., van Kreveld, M., Overmans, M. and Schwarzkopf, O., *Computational Geometry: Algorithms and Applications*, 2nd revised ed., Springer-Verlag, Berlin (2000).
[20] Chan, T.M. "Output-sensitive results on convex hulls, extreme points and related problems", *ACM Symposium on Computational Geometry*, (1995).
[21] McQueen, M.M. and Toussaint, G.T. "On the ultimate convex hull algorithm in practice", *Pattern Recognition Letters*, 3(1), pp. 29–34 (1985).
[22] Chazelle, B. "On the convex layers of a planar set", *IEEE Transactions on Information Theory*, IT-31, pp. 509–517 (1985).
[23] Deb, K. and Saxena, D.K. "On finding pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems", KanGAL Report No. 2005011, IIT, Kanpur (2005).
[24] Miettinen, K., *Nonlinear Multiobjective Optimization*, Kluwer, Boston (1999).
[25] Konak, A., Coit, A.W. and Smith, A.E. "Multi-objective optimization using genetic algorithms: a tutorial", *Reliability Engineering and System Safety*, 91(9), pp. 992–1007 (2006).
[26] Khare, V., Yao, X. and Deb, K. "Performance scaling of multi-objective evolutionary algorithms", KanGAL Report No. 2002009, pp. 1–15 (2002).
[27] Deb, K., Agrawal, S., Pratap, A. and Meyarivan, T. "A fast elitist non-dominated sorting genetic algorithm for multi-objective: NSGA-II", In *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pp. 846–858, Springer-Verlag (2000).
[28] Deb, K. and Goel, T. "Controlled elitist non-dominated sorting genetic algorithms for better convergence", in: *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization, EMO*, pp. 67–81 (2001).
[29] Jensen, M.T. "Reducing the run-time complexity of multi-objective EAs: the NSGA-II and other algorithms", *IEEE Transaction on Evolutionary Computation*, 7(5), pp. 503–515 (2003).
[30] Liu, L., Neittaanmaki, P. and Krizek, M. "Second-order optimality conditions for non-dominated solutions of multiobjective programming with $C^{1,1}$ data", *Applications of Mathematics*, 45(5), pp. 381–397 (2000).
[31] Bentley, J.L. "Multidimensional divide-and-conquer, communication", *ACM*, 23(4), pp. 214–229 (1980).
[32] Homaifar, A., Lai, S.H.-V and Qi, X. "Constrained optimization via genetic algorithms", *Simulation*, 62(4), pp. 242–254 (1994).
[33] Michalewicz, Z. and Schoenauer, M. "Evolutionary algorithms for constrained parameter optimization problems", *Evolutionary Computation Journal*, 4(1), pp. 1–32 (1996).

Mansoor Davoodi Monfared received his B.S. degree in Computer Science from Vali-Asr University, Rafsanjan, Iran, in 2005, and an M.S. degree from Amirkabir University in 2007. Currently, he is studying towards Ph.D. degree in the Computer Science Department of Amirkabir University. His main research areas include computational geometry, imprecise data modeling and multi-objective optimization.

Ali Mohades is Manager of the Computer Science group in the Department of Mathematics and Computer Science at Amirkabir University of Technology, Tehran, Iran, where he is also Dean of the Algorithms and Computational Geometry Laboratory. His main fields of interest are computational geometry, motion planning and facility location.

Jafar Rezaei received his M.S. degree in Operations Research from Tehran University, Iran, in 2002. He is currently a Ph.D. student in the Faculty of Technology, Policy and Management, at Delft University of Technology in the Netherlands. He has published more than 30 research papers in several international conferences and peer-reviewed journals, such as the International Journal of Production Economics, the International Journal of Production Research and Applied Mathematical Modelling. His research interests include multi-objective and multi-criteria decision-making in supply chain management.