An Efficient Chemical Reaction Optimization Algorithm for Multiobjective Optimization

Slim Bechikh, Abir Chaabani, and Lamjed Ben Said

***Abstract*—Recently, a new metaheuristic called chemical reaction optimization was proposed. This search algorithm, inspired by chemical reactions launched during collisions, inherits several features from other metaheuristics such as simulated annealing and particle swarm optimization. This fact has made it, nowadays, one of the most powerful search algorithms in solving mono-objective optimization problems. In this paper, we propose a multiobjective variant of chemical reaction optimization, called nondominated sorting chemical reaction optimization, in an attempt to exploit chemical reaction optimization features in tackling problems involving multiple conflicting criteria. Since our approach is based on nondominated sorting, one of the main contributions of this paper is the proposal of a new quasi-linear average time complexity quick nondominated sorting algorithm; thereby making our multiobjective algorithm efficient from a computational cost viewpoint. The experimental comparisons against several other multiobjective algorithms on a variety of benchmark problems involving various difficulties show the effectiveness and the efficiency of this multiobjective version in providing a well-converged and well-diversified approximation of the Pareto front.**

***Index Terms*—Chemical reaction optimization, evolutionary computation, multiobjective optimization, nondominated sorting.**

I. Introduction

MOST real-world optimization problems encountered in practice have a multiobjective nature. They involve several conflicting and incommensurable objectives to be maximized or minimized simultaneously with respect to some constraints. The resolution of a multiobjective optimization problem (MOP) gives rise to a set of compromise solutions presenting the optimal trade-offs between the different objectives. When plotted in the objective space, the set of compromise solutions is called the Pareto-optimal front. Several methods were proposed in the specialized literature in order to approximate the Pareto-optimal front for the discrete case and the continuous one [51]. Mimicking the principles of biological evolution, evolutionary algorithms (EAs) have earned popularity in solving MOPs during the two last decades and beyond thanks to: 1) their ability to provide a set of compromise solutions as output on a single simulation run and 2) their insensitivity to the geometrical features of the objective functions such as nonconvexity, discontinuity, nonuniformity of the search space, etc [51]. As a consequence EAs Multi-Objective Evolutionary Algorithm (MOEAs) in handling MOPs, a new branch in the optimization research field has appeared. This branch is called evolutionary multiobjective optimization (EMO) and is actually one of the most attractive and active research fields in computer science. Several fitness assignment schemes were proposed in the EMO literature. These schemes could be classified into three families as follows.

Manuscript received May 6, 2014; revised August 12, 2014 and October 13, 2014; accepted October 14, 2014. Date of publication October 30, 2014; date of current version September 14, 2015. This paper was recommended by Associate Editor H. Ishibuchi.

The authors are with the Stratégies d'Optimisation et Informatique Intelligente (SOIE) Laboratory, ISG-Tunis, University of Tunis, Tunis 8000, Tunisia (e-mail: slim.bechikh@gmail.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2014.2363878

A. Pareto Ranking-Based Schemes

These schemes make a direct use of the Pareto dominance relation [4] in designing the selection rules that allows distinguishing between the different population individuals. The first attempt to use Pareto dominance fitness assignment that successfully generates a set of nondominated points is by Fonseca and Fleming [4]. Their algorithm was called multiobjective genetic algorithm (MOGA) which was shortly followed by niched Pareto genetic algorithm (NPGA) presented by Horn *et al.* [7]. Both MOGA and NPGA have shown the effectiveness and efficiency of such fitness assignment schemes and were followed by a number of more advanced MOEAs that further improved upon these early designs. Some of the most notable of them are nondominated sorting genetic algorithm (NSGA) [8], NSGA-II [9] which is an improved version of the former, strength Pareto EA (SPEA) [10], its improved version SPEA2 [11], and Pareto envelope selection algorithm [12].

B. Indicator-Based Schemes

More recently, in 2004, Zitzler and Künzli [13] proposed a new fitness assignment schema based on the use of performance indicators that are compliant with the Pareto dominance relations. The most used indicators are the hypervolume [14], the epsilon indicator and the R2 one [15]. The basic idea is to evaluate an individual in terms of the loss in quality indicator if that individual is deleted from the population. Consequently, solutions having smallest losses are de-emphasized to remain in the race for the next generations; however, solutions having large losses are emphasized in mating selection and the environmental one. Some representative algorithms falling within this class are indicator-based EA (IBEA) [16], S-metric selection EMO algorithm [20], R2 EMO algorithm (R2EMOA) [21], and approximation-guided EMO (AGE) [43].

C. Decomposition-Based Schemes

Decomposition-based algorithms decompose the MOP into a number of scalar optimization sub-problems and optimize

them simultaneously. Each sub-problem is optimized by only using information from its neighboring sub-problems. The most cited algorithms falling within this family are multiobjective genetic local search [19], NSGA-III (a many-objective optimization method based on the NSGA-II) [5], [6] and MOEA based on decomposition (MOEA/D) [20], [27], [29]. Actually, decomposition seems to be an interesting approach to tackle many-objective problems, since it takes into account not only the number of improvements of one solution over another but also the quantities of these improvements.

Chemical reactions possess several characteristics and efficient components, states, process, and events that can be designed as an optimization method. Recently, a new metaheuristic called chemical reaction optimization (CRO) was proposed by Lam and Li [22]. The first version was proposed for combinatorial problems. After that, it was followed by a second one for continuous problems [23]. There are other works inspired from chemical reaction were proposed in the literature. We mention for instance an heuristic based computational algorithm called artificial chemical reaction optimization algorithm for global optimization (ACRO) proposed by Alatas [44]. In fact, the CRO algorithm has demonstrated its effectiveness and efficiency in solving different single-objective real-world and benchmark problems [24], [25]. The CRO algorithm has several nice features (see Section II-B) rendering it one of the most efficient metaheuristics for mono-objective optimization. For this reason, we propose in this paper the first multiobjective CRO algorithm in attempt to exploit the nice CRO characteristics in solving MOPs. The main contributions of this paper are the following.

1) Proposing, for the first time, a multiobjective CRO that we call nondominated sorting CRO (NCRO) for solving MOPs.
2) Proposing a new Non-Dominated Sorting Algorithm (NDSA), called quick-NDSA (Q-NDSA), having a quasi-linear average time complexity while using a linear data structure and not a nonlinear one.
3) Showing that our NCRO has a lower average time complexity than several other MOEAs.
4) Demonstrating the outperformance of our NCRO over some of the most representative MOEAs that are NSGA-II, IBEA, MOEA/D, and AGE on different commonly used benchmark problems from the Deb-Thiele-Laumanns-Zitzler (DTLZ) suite and the walking fish group (WFG) one.

II. Related Works

A. Multiobjective Optimization

A MOP consists of minimizing or maximizing an objective function vector under some constraints. The general form of a MOP is as follows [2]:

$$\begin{cases} \text{Min } f(x) = & \left[f_1(x), f_2(x), \ldots, f_M(x)\right]^T \\ g_j(x) \geq 0 & j = 1, \ldots, P \\ h_k(x) = 0 & k = 1, \ldots, Q \\ x_i^L \leq x_i \leq x_i^U & i = 1, \ldots, n \end{cases} \quad (1)$$

where M is the number of objective functions, P is the number of inequality constraints, Q is the number of equality constraints, and x_i^L and x_i^U correspond to the lower and upper bounds of the variable x_i. A solution satisfying the $(P + Q)$ constraints in addition to the bound constraints defined in (1) is said feasible and the set of all feasible solutions defines the feasible search space denoted by X. In this formulation, we consider a minimization MOP, since maximization can be easily turned to minimization based on the duality principle. The resolution of a MOP yields a set of trade-off solutions, called Pareto optimal solutions or nondominated solutions, and the image of this set in the objective space is called the Pareto-optimal front. Hence, the resolution of a MOP consists in approximating the whole Pareto front. In the following, we give some background definitions related to multiobjective optimization.

Definition 1: (Pareto optimality) A solution x^* is Pareto optimal if there does not exist any $x \in X$ such that $f_m(x) < f_m(x^*)$ for all $m \in I$ where $I = \{1, 2, \ldots, M\}$.

The definition of Pareto optimality states that x^* is Pareto optimal, if no feasible vector x exists which would improve some objective without causing a simultaneous worsening in at least another one. Other important definitions associated with Pareto optimality are essentially the following.

Definition 2: (Pareto dominance) A solution $u = (u_1, u_2, \ldots, u_n)$ is said to dominate another solution $v = (v_1, v_2, \ldots, v_n)$ denoted by $f(u) \preceq f(v)$ if and only if $f(u)$ is partially less than $f(v)$. In other words, $\forall m \in \{1, \ldots, M\}$, we have $f_m(u) \leq f_m(v)$ and $\exists m \in \{1, \ldots, M\}$ where $f_m(u) < f_m(v)$.

Definition 3: (Pareto optimal set) For a MOP $f(x)$, the Pareto optimal set is $P^* = \{x \in X \neg \exists x' \in X, f(x') \preceq f(x)\}$. In the classical multiobjective optimization literature, this set is called the efficient set.

Definition 4: (Pareto optimal front). For a given MOP $f(x)$ and its Pareto optimal set P^*, the Pareto-optimal front is $PF^* = \{f(x), x \in P^*\}$.

MOPs have received considerable attention in operations research [26]. Traditional methods to solve MOPs transform the MOP to a SOP via an aggregation method. Several aggregative methods were proposed in the specialized literature. We cite, for example, the weighted-sum method, the ε-constraint method, the reference direction method, the light beam search approach, and so forth (see [3] for an exhaustive review). The main problem with these methods is that they require several parameters to tune subjectively by the decision making which is not an easy task. Additionally, these methods depend on the geometrical features of the objective functions and they can provide only one solution in each simulation run. Due to their population-based nature, over the two last decades, MOEAs have been demonstrated to be effective black-box tools to handle MOPs (see [28] for a detailed recent survey).

B. CRO: Basic Algorithm and Features

CRO is a new recently proposed metaheuristic [22]. It is inspired by the first two laws of thermodynamics of

chemical reactions. CRO evolves a population of molecules by means of four chemical reactions: 1) On-wall ineffective collision; 2) decomposition; 3) intermolecular ineffective collision; and 4) synthesis. Consequently, similarly to genetic algorithm (GA), the molecule corresponds to the population individual and chemical reactions correspond to the variation operators. However, CRO is distinguished by the fact that environmental selection is performed by the variation operator. Differently to GA which generates an offspring population then makes a competition between the latter and the parent population, in CRO once an offspring is generated, it competes for survival with its parent(s) during the realization of the corresponding chemical reaction. Similar to other evolutionary algorithms or metaheuristics, CRO consists of three stages: 1) initialization; 2) iterations; and 3) the final stage. The first one begins by initializing the different parameters that are as follows.

1) *PopSize:* The molecule population size.
2) *KELossRate:* The loss rate in terms of kinetic energy (KE) during the reaction.
3) *MoleColl:* A parameter varying between 0 and 1 deciding whether the chemical reaction to be performed is uni-molecular or mutli-molecular.
4) *Buffer:* The initial energy in the buffer.
5) *InitialKE:* The initial *KE* energy.
6) α and β*:* Two parameters controlling the intensification and diversification.

For more details about the role of each of these parameters, the interested reader is invited to confer to [23] and [30]. Once the initialization set is performed, the molecule population is created and the evolution process begins. The latter is based on the following four variation operators (elementary chemical reactions).

1) *On-wall ineffective collision:* This reaction corresponds to situation when a molecule collides with a wall of the container and then bounces away remaining in one single unit. In this collision, we only perturb the existing molecule structure ω to ω'. This could be done by any neighborhood operator $N(\bullet)$.
2) *Decomposition:* It corresponds to the situation when a molecule hits a wall and then breaks into several parts (for simplicity, we consider two parts in this paper). Any mechanism that can produce ω_1' and ω_2' from ω is allowed. The goal of decomposition is to allow the algorithm to explore other regions of the search space after enough local search by the ineffective collisions.
3) *Intermolecular ineffective collision:* This reaction takes place when multiple molecules collide with each other and then bounce away. The molecules (assume two) remain unchanged before and after the process. This elementary reaction is very similar to the uni-molecular ineffective counterpart since we generate ω_1' and ω_2' from ω_1 and ω_2 such that $\omega_1' = N(\omega_1)$ and $\omega_2' = N(\omega_2)$. The goal of this reaction is to explore several neighborhoods simultaneously each corresponding to a molecule.
4) *Synthesis:* This reaction is the opposite of decomposition. A synthesis happens when multiple (assume two) molecules hit against each other and fuse together. We obtain ω' from the fusion of ω_1 and ω_2. Any mechanism allowing the combination of solutions is allowed, where the resultant molecule is in a region farther away from the existing ones in the solution space.

The CRO mechanism is based on energy distribution, therefore, molecules with energy move and trigger collisions. A molecule can either hit on a wall of the container or collide with each other. This is decided by generating a random number t in [0, 1]. If $t > MoleColl$ or the system only has one molecule, we have a uni-molecular collision. Otherwise, an intermolecular collision follows. The two ineffective collisions implement local search (intensification) while decomposition and synthesis give the effect of diversification. The two former collisions get new molecular structures in their own neighborhoods on potential energy surface (PES) (i.e., they pick new solutions close to the original ones). Thus, the PE of the resultant molecules tends to be close to those of the original ones. Conversely, decomposition and synthesis tend to obtain new molecular structures which may be far away from their immediate neighborhoods on PES. This allows making a good trade-off between exploitation and exploration. The algorithm undergoes these different reactions until the satisfaction of the stopping criteria. After that, it outputs the best solution found during the overall chemical process. It is important to note that the synthesis operation decreases the population size. Contrariwise, the decomposition increases it. The molecule in CRO has several attributes, some of which are essential to the basic operations, that is: 1) the molecular structure ω expressing the solution encoding of the problem at hand; 2) the potential energy (PE) corresponding to the objective function value of the considered molecule; and 3) the KE corresponding to nonnegative number that quantifies the tolerance of the system accepting a worse solution than the existing one [similarly to simulated annealing (SA)]. The optional attributes are as follows.

1) *Number of Hits (NumHit):* Is a record of the total number of hits a molecule has taken.
2) *Minimum Structure* (*MinStruct*)*:* It is the ω with the minimum corresponding PE which a molecule has attained so far. After a molecule experiences a certain number of collisions, it has undergone many transformations of its structure, with different corresponding PE. *MinStruct* is the one with the lowest PE in its own reaction history. This attribute is very similar to the concept of local best in particle swarm optimization (PSO).
3) *Minimum* PE (*MinPE*)*:* When a molecule attains its *MinStruct*, *MinPE* is its corresponding PE.
4) *Minimum Hit Number (MinHit):* It is the number of hits when a molecule realizes *MinStruct*. It is an abstract notation of time when *MinStruct* is achieved. For more details about the role of each of these attributes, the reader is invited to refer to [22].

The CRO has been recently applied successfully to different combinatorial and continuous optimization problems [31]–[33]. Several nice properties for the CRO have been detected. These properties are as follows.

1) The CRO framework allows deploying different operators to suit different problems.

2) Energy conversion and energy transfer in different entities and in different forms make CRO unique among metaheursitics.
3) Other attributes can easily be incorporated into the molecule. This gives flexibility to design different operators.
4) CRO enjoys the advantages of SA and GA (and sometimes PSO when using the *MinStruct* option). It can be even seen as a hyper-heuristic since it controls the variation operations to be realized.

C. Brief Summary of Existing NDSAs

Since one of the main contributions of this paper is the proposal of a new quasi-linear average time complexity NDSA, we give in this section a brief summary of existing work in the nondominated area. The goal of any NDSA is to classify a population of solutions into nondominated fronts where: 1) two solutions belonging to the same front are nondominated with each other and 2) each solution belonging to front F_{k+1} is dominated by at least one solution from F_k. Solutions having better front ranks are preferred over solution having worse ones. The number of existing NDSAs in the literature is not large. These algorithms could be classified into two main families.

1) Pareto Dominance Decomposition-Based Algorithms: The basic idea is to find the Pareto rank of each solution progressively by considering initially the first objective, then the second one, and so on. Since, we did not use all objectives simultaneously but rather sequentially to find the Pareto ranks, we can say that these algorithms decompose the Pareto dominance relation. The two main works in this family are [34] and [35]. They both independently extend Kung *et al.*'s [36] divide and conquer-based vector sorting algorithm which identifies only the first front. Based on Kung's study, this type of algorithms has a time complexity of $N(log(N))^{M-1}$ (where M is the number of objectives and N is the population size). It is important to note that Jensen's algorithm puts duplicates solutions in the same front which represents the major drawback for this algorithm.

2) Pareto Dominance Evaluation-Based Algorithms: These algorithms finds the Pareto ranks based on the overall evaluation of the Pareto dominance between pairs of solutions (i.e., by considering all objectives simultaneously). Algorithms falling in this family could be categorized into two subfamilies.

1) *Linear Data Structure-Based Algorithms:* In this family, we find mainly Deb *et al.*'s [9] NDSA where for each of the N solutions, two quantities are computed: 1) n_i corresponding to the number of solutions dominating solution i and 2) s_i corresponding to set of solutions dominated by solution i. The calculation of these two entities requires $O(MN^2)$ comparisons. After that, solution having $n_i = 0$ are identified and saved in a list named F_1. The latter is called the current front. Now, for each solution i in the current front, each member k from its set s_i is visited and its n_k count is reduced by one. In doing so, if for any member k the count n_k becomes zero, we put it in a separate list H. Once all members of the current front have been checked, we declare the solutions in the list F_1 as solutions of the first front. We then continue this process using the newly identified front H as our current front. Each such iteration requires $O(N)$ computations. This process continues till all fronts are identified. Since at most there can be N fronts, the worst case complexity of this loop is $O(N^2)$. Hence, the overall time complexity of Deb *et al.*'s [9] NDSA is $O(MN^2) + O(N^2) = O(MN^2)$.
2) *Nonlinear Data Structure-Based Algorithms:* The basic idea is to exploit nonlinear data structures coupled with the divide-and-conquer strategy in order to reduce the number required comparisons for identifying the different fronts. We find three recently proposed algorithms in this family: 1) Fang *et al.*'s [37] NDSA; 2) the deductive sort; and 3) the climbing sort. The two latter are proposed by McClymont and Keedwell [38]. Fang *et al.* [37] NDSA uses a tree-like data structure, called dominance tree, in order to save the dominance information among solutions with the aim to reduce the number of repetitive (redundant) comparisons. On the one hand, a solution of a child node in the dominance tree cannot dominate any of its parent's siblings. On the other hand, the solution of a node may not necessarily dominate its siblings' children. The authors demonstrated that the best case of their algorithm corresponds to the situation where each front contains a unique solution; however, the worst one corresponds to the case where all solutions are nondominated with each others (they form a single front). The worst case complexity of Fang *et al.*'s [37] algorithm is demonstrated to be $O(N^2)$. The best case one is shown to be $O(MNlog(N))$. The average case is demonstrated to be $O((1/p)MNlog(N))$ where p is the probability that either of two solutions dominate the other. Since the p decreases during the MOEA run and with the increase of the number of objectives, the general time complexity of this algorithm remains near $O(N^2)$. The deductive sort and the climbing one are based on a graph-like data structure called dominance graph. The latter is used to save the direct dominance, indirect dominance, and the mutual nondominance relationships between the different solutions. In this way, the algorithms using such data structures avoid not only redundant comparisons but also useless ones. The deductive sorting and the climbing one have the same worst case and best case as for Fang *et al.*'s [37] NDSA. Although, the number of comparisons is demonstrated theoretically and experimentally to be reduced when compared to Deb *et al.*'s [9] NDSA, the overall time complexity for each of these algorithms remains nonlinear with an order of $O(N^2)$.

III. NCRO for Multiobjective Optimization

A. NCRO Basic Scheme

This section gives the main algorithmic scheme of NCRO which is illustrated by Fig. 1. The step-by-step procedure shows that our NCRO algorithm has the same algorithmic

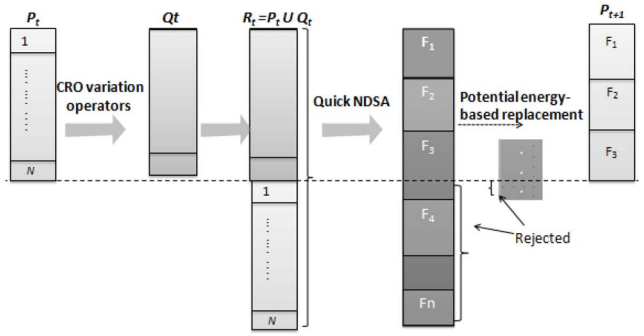

Fig. 1. NCRO basic iteration.

scheme of NSGA-II with some modifications. The NCRO algorithm proceeds as follows. First, an offspring population Q_t is obtained by applying the CRO variation operators. Second, since CRO controls the moves in the search space through its energy management laws [22], the Q_t population should be updated. Therefore, some adaptations should be done to fit the CRO to the NSGA-II scheme. These adaptations are mainly: 1) the PE assignment and 2) the offspring generation, which will be detailed in the next subsections. Once, the offspring population is generated and updated, a combined population $R_t = P_t \cup Q_t$ is formed. The population size of R_t is larger than the predefined population size N, since both parent and offspring population members are included in R_t. As a consequence, elitism is ensured. It is important to note that the R_t size may be lower than $2N$, since the updated procedure of CRO can reject nonpromising solutions. After this step, the R_t population is sorted according to nondomination criterion using our new Q-NDSA algorithm (presented in Section II-B). Next, we will fill the new population P_{t+1} with the best solutions in R_t. Therefore, solutions having smallest fitness value (i.e., PE value) are the best solutions in the combined population. For this, we calculate the PE value of solutions belonging to R_t, based on the Pareto rank and the crowding distance measure. We note that, when computing the crowding factor which is a front-wise process, solutions having a crowding distance of zero in NCRO population are deleted and replaced by randomly generated ones having different objective vectors. In fact, a solution having a crowding factor of zero is a solution that is duplicated in the objective space with at least M other solutions from the same front where M is the number of objectives. Although this case is extremely seldom to be produced, we should handle it since we need to ensure that our PE assignment scheme is Pareto dominance preserving for all possible cases (see Theorem 1 proof). Once R_t population members are evaluated, we choose the N best solutions in R_t to fill the new population P_{t+1}. The new population of size N is used now in the following iteration steps to create a new offspring population Q_{t+1}. We give the pseudocode of NCRO in Appendix. As well, the NCRO Java source code could be obtained via the following link: https://sites.google.com/site/slimbechikh/sourcecodes.

1) PE Assignment: For the mono-objective case, the objective function of a problem can be used as a fitness function evaluating directly the quality of molecules. Nevertheless, in the multiobjective case, the quality of a molecule cannot be accessed directly in the same manner. For this reason, we should adapt the PE concept when considering several objectives simultaneously. In fact, we propose a new formula for the PE that is based on the Pareto rank and crowding distance measure. The PE energy formula is given by

$$\mathrm{PE}(x) = \mathrm{rank}(x) + e^{(-\mathrm{crowd}(x))} \tag{2}$$

where x is a decision vector $(x_1, x_2, \ldots, x_n)$, rank is the Pareto rank of a solution and crowd is a measure corresponding to the crowding distance used in NSGA-II [9]. In what follows, we study the main characteristic of our PE formula through Theorem 1.

Theorem 1: The PE given by (2) preserves the order induced by the Pareto dominance relation.

Proof: The crowd value is always a positive number belonging to $]0, +\infty[$ as well, $\lim_{\mathrm{crowd}(x)\longrightarrow 0} e^{(-\mathrm{crowd}(x))} = 1$ and $\lim_{\mathrm{crowd}(x)\longrightarrow +\infty} e^{(-\mathrm{crowd}(x))} = 0$. Therefore, the $e^{(-\mathrm{crowd}(x))}$ quantity belongs to the $[0, 1]$ interval. It is important to note that the PE value has to be minimized. Now, we show that our PE formula preserves the Pareto dominance relation order: ■

1) *Case 1:* Consider 2 solutions $A, B \in X$ with the Pareto rank of A is better (i.e., lower) than B one. In this situation the PE value of A is lower than B one since $\mathrm{rank}(A) + q < \mathrm{rank}(B) + q'$ where $q = e^{(-\mathrm{crowd}(A))}$, $q' = e^{(-\mathrm{crowd}(B))}$ and $q, q' < 1$.
2) *Case 2:* The Pareto rank of B is better than A one. Similarly to case 1, we can derive that the PE value assigned to solution B is lower than the value assigned to A.
3) *Case 3:* A and B belonging to the same front. From the exponential function property we know that as the crowd of a solution i is larger, the $e^{(-\mathrm{crowd}(i))}$ will be lower. Consequently, the solution located in a lesser crowded region takes the better value (i.e., minimal one). In this respect, we can say that this formula can guide the selection process toward a uniformly spread out Pareto-optimal front. As a result, this fitness scheme respects the NSGA-II crowding operator order.

In summary, we can properly say that our PE formula preserves the Pareto dominance relation.

2) Offspring Population Generation in NCRO: CRO is distinguished from other meta-heuristics by the fact that environmental selection is performed within the variation operators. Once an offspring is generated, it competes for survival with its parent(s) during the realization of the corresponding chemical reaction. Therefore, our second adaptation consists in relaxing the CRO verification in order to avoid the replacement at first place. Once the offspring population Q_t is generated, we apply then the energy management laws of CRO to update the Q_t population and determine which solutions (i.e., authorized moves) should remain in the update offspring population. To do this, we mark for each molecule, the number of offspring generated, its child and the collision type performed, to be able then to apply correctly the CRO energy management rules. In this way, the product molecules exist together with the reactant molecules. To correct this fact, we apply

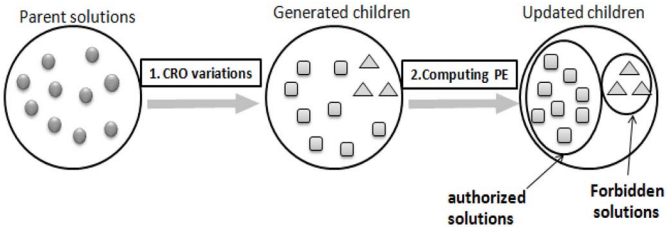

Fig. 2. Controlling the moves in NCROs offspring generation step.

for each reactant molecules the corresponding energy management laws. Thus, the reactants or children that do not obey this law will be removed from the offspring population and then any product molecules cannot exist together with its reactant molecules as well the energy transforming between different molecules will be applied correctly based on the basic law of CRO. Consequently, the total energy in the system remains constant. Fig. 2 shows the adaptation idea. It looks like we control moves in the search space by classifying them into: 1) authorized moves and 2) forbidden ones. The former one corresponds to the movements which obey the PE management rules of such operator and the latter corresponds to the contrary case. In this respect, our NCRO has the same algorithmic scheme than NSGA-II and as well it respects the energy management laws of CRO.

B. Q-NDSA

1) Q-NDSA for the Bi-Objective Case: The bi-objective case of Q-NDSA is inspired by Jensen [34], Yukish [35], and Fang *et al.* [37]. The basic idea of Q-NDSA for the bi-objective case is as follows. Firstly, all population members are sorted based on the first objective f_1. In this way, solutions preceding a particular solution u may dominate it. However, solutions following u cannot dominate it. Consequently, the first solution (having the minimal f_1 value) is assigned a Pareto rank of 1, since it is not dominated with respect to all population members. Once the first nondominated solution is identified, the next step is to find the rest of nondominated solutions. In order to achieve this goal, we stock $f_2(u)$ in a witness variable W and we visit the second solution v according to the sorting based on f_1. If $(f_2(v) < W)$ then v is assigned a rank of 1 since it is not dominated by u, and W takes $f_2(v)$. This process is repeated for every solution of the rest of the population to find all members of the first nondominated level in the population. Once the first front F_1 is identified, we discard it temporary from the population P and we reiterate the overall process with the aim to find next front F_2, and so on. The algorithm repeats this process until the classification of all population members into nondominated fronts. This fact can considerably reduce the number of comparisons and then can improve the computational efficiency of any Pareto dominance evaluation-based algorithms. Fig. 3 shows an example of execution of the bi-objective sorting.

In what follows, we study the computational cost of our Q-NDSA for the bi-objective case from a time viewpoint: the best case of our algorithm corresponds to the case where all population members are nondominated with each others,

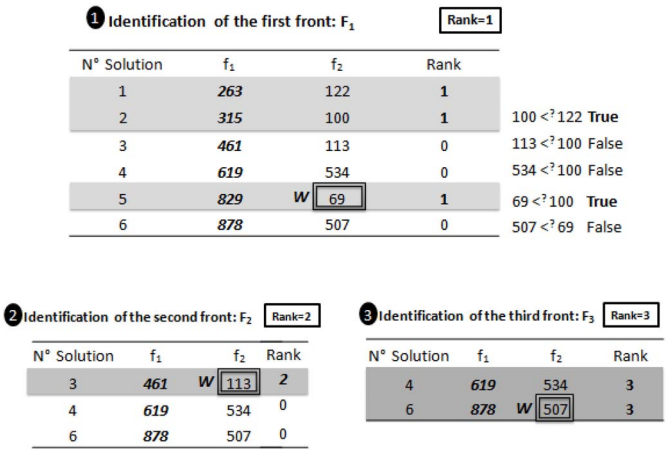

Fig. 3. Running example of Q-NDSA for the bi-objective case.

and therefore they form a single front. For this case, we require $Nlog(N)$ computations to sort the population using quick-sort and $(N-1)$ comparisons with the witness variable W, in order to conclude that all members are nondominated. Thus, the overall time complexity for the best case is $O(Nlog(N) + (N-1)) \sim O(Nlog(N))$. The worst case corresponds to the situation where each front is composed by a single solution. In this way, we have N fronts. For this case, we require $Nlog(N)$ computations to sort the population using quick-sort and several comparisons to find the fronts. Therefore, in each iteration, one solution will be discarded from the population. The first front requires $(N-1)$ comparisons. The second one requires $(N-2)$, the third one necessitates $(N-3)$, and so on. The last front requires 0 comparison. The overall number of comparisons for finding all fronts is $Z = (N-1) + (N-2) + \cdots + 0$. We remark that Z corresponds to an arithmetic suite with a common difference of 1 and N terms. Consequently, $Z = (N)[(N-1)+0)]/2 \sim N^2$. Hence, the overall computational cost for the worst case is $O(Nlog(N) + Z) \sim O(N^2)$. The average case corresponds to obtaining number of fronts (NF) fronts where $NF <<< N$ (NF is too small compared to N). Consequently, we need as well $Nlog(N)$ computations for quick-sort and Y comparisons for the detection of the fronts. Assuming q to be the average size of a front, $Y = ((N-1)-q) + ((N-1)-2q) + \cdots + 0$. Thus, the overall complexity for the average case is $O(Nlog(N) + Y) \sim O(Nlog(N) + N) \sim O(Nlog(N))$.

In summary, the Q-NDSA for the bi-objective case has the following properties: 1) a quasi-linear time complexity of $Nlog(N)$ for the best case; 2) a quasi-linear time complexity of $Nlog(N)$ for the average case; and 3) a quadratic time complexity of N^2 for the worst case.

2) Q-NDSA for the M-Objective Case: The basic idea of the Q-NDSA for the M-objective case is as follows. Once nondominated solutions based on the two first objectives are identified, we apply a rank correction strategy in order to find the rest of nondominated solutions for the M-objective case (see Fig. 4). In fact, solutions having a rank of 1 based on f_1 and f_2 remain always nondominated. However, solutions dominated based on f_1 and f_2 may become nondominated when

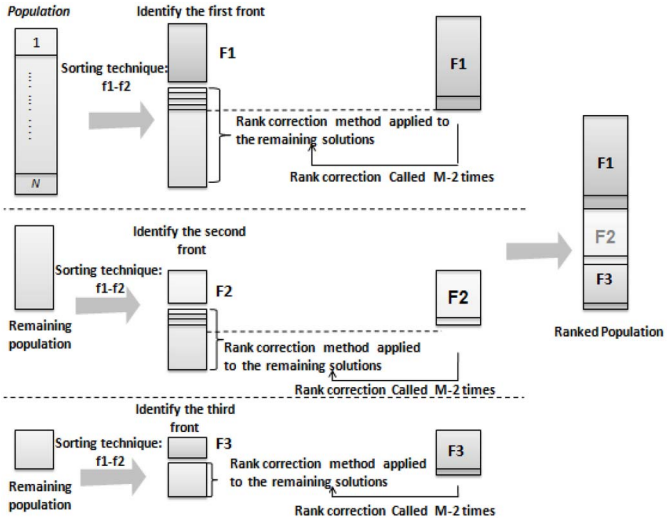

Fig. 4. Illustration of the Q-NDSA scheme.

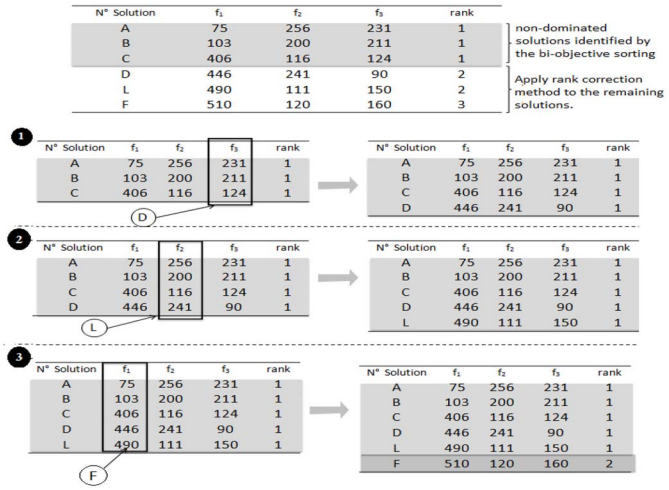

Fig. 5. Running example of the rank correction procedure for the tri-objective case.

considering a new objective. In Fig. 4, we remark that the size of such front F is small at the beginning and then it increases when nondominated solutions are detected by the rank correction method. The main part of the Q-NDSA is given by Algorithm 2 which mainly follows the scheme of Fig. 4.

a) Rank correction procedure: As we mentioned before, the rank correction procedure corrects the rank of solutions when considering M objectives together. To explain the basic idea of this procedure, we assume the tri-objective case. Firstly, all nondominated members form a set called E and are sorted based on the last objective f_3. Secondly, a binary search is performed according to f_3 to determine if a solution $u \in \{P \setminus E\}$ is nondominated. In this way, if u precedes all members in E, then u is a nondominated solution. Otherwise, u may be dominated by some members q belonging to the set E. For this, a binary search and a precedence check out are performed according to f_2. If u precedes all q members, then u is a nondominated solution. Otherwise, the same process is repeated according to f_1 to determine definitively if u is a dominated or nondominated solution. Fig. 5 illustrates an example of the tri-objective case. We have in this figure three

Algorithm 1 Q-NDSA Pseudocode

Input: Solution Set P
Output: Ranked P
01: **Begin**
02: $P \leftarrow$ QuickSort (P, 1) /* Sort P based on the first objective f_1*/
03: $Rank \leftarrow 1$
04: F = ϕ
05: **While** ($P \neq \phi$) **do**
06: /*Step 1: Identifying the actual best front from (truncated) P*/
07: /**Step 1.1: Identifying the first non-dominated solutions**/
08: $W \leftarrow s_1$.ObjValue(2) /* S_1 is the current solution having the minimal f_1 value in truncated P */
09: **For** each $s_i \epsilon P$ **do**
10: **If** (s_i.ObjValue (2) $<W$) **Then** /*S_i is not dominated by S_1*/
11: $W \leftarrow si$.ObjValue(2)
12: $s_i.Rank = Rank$
13: $F \leftarrow$Union (F, s_i) /* Store the solutions belonging to the same front in a temporary variable F*/
14: $P \leftarrow$Remove (s_i, P) /* Remove s_i from (truncated) P*/
15: **End If**
16: **End for**
17: /**Step 1.2: Identifying the rest of non-dominated solutions via rank correction**/
18: **If** ($M > 2$) **Then** /*The correction step is required only when $M>$ 2*/
19: $m \leftarrow 3$
20: **while** (($m \leq M$) *and* ($p \neq 0$)) **do**
21: **RankCorrection** (F, $F.Rank$, P, m)
22: $m \leftarrow m + 1$
23: **End while**
24: **End If**
25: /*Step 2: Increment the actual Pareto rank to identify the next front*/
26: $Rank \leftarrow Rank + 1$
27: **End while**
28: **End**

solutions A, B, C which form the first part of the front F_1, i.e., E. In order to find the rest of F_1, we proceed as follows. The solution D, is tested firstly according to f_3. This test result shows that D is a nondominated solution having the minimal f_3 value. After, we proceed for the solution L. Similarly, we test it according to f_3, we find that L can be dominated by the solution C and D. So thus, we test it according to f_2 with respect to C and D only. In this way, we deduce that solution L is a nondominated one. Consequently, we put it in F_1. The solution F, is compared now with respect the updated front $\{\{E\}, D, L\}$, firstly according to f_3, then f_2 and at last according to f_1. As a result, this precedence checks out shows that F is dominated by solutions C and E. The overall process is repeated when we add a new objective. In general, for M objectives, we execute this procedure $(M-2)$ times. Once the front F_1 is definitively identified, it will be discarded temporary from the population and we reiterate the overall process with the aim to find the next front F_2 and so on.

In summary, we conclude that the execution of the rank correction method can face several cases.

1) The best case is when we apply the binary search according to one objective only.
2) The average case is when we perform the binary search NV times where $1 < NV < M$.
3) The worst case corresponds to the situation where we apply the binary search M times.

The rank correction procedure is illustrated by the Algorithm 3. In what follows, we study the computational cost of the rank correction procedure from a time viewpoint. Start with the best case, we need ($|E|log(|E|)$) computations

Algorithm 2 RankCorrection Procedure Pseudocode

Input: Solution Set E, Integer *ERank*, Solution Set P and Integer *ObjIndex*
Output: Other identified non-dominated solutions from the truncated population P

```
01: Begin
02: E ← QuickSort (E, ObjIndex) /*Sort the first part of the current front
    according to current objective ObjIndex */
03: For each s from P do /*Remark:P is already reduced in Quick_NDSA
    routine */
04:     T = false /*T is a boolean variable*/
05:     While (!T) do
06:         position ← BinarySearch(s, E, ObjIndex)
07:         PDS ← Precedence(position, E, ObjIndex) /*identify the
            q solutions that can dominate s and put them in PDS*/
08:         If (PDS = Ø) Then /*s is definitively a non-dominated with
            respect to E */
09:             E ← BinaryInsertion(s, E)/*add s to the current front */
10:             s.rank ← ERank
11:             T ← true
12:         Else If (ObjIndex> 1) Then
13:                 E ← PDS
14:                 ObjIndex = ObjIndex − 1
15:         Else /*ObjIndex=1*/
16:                 T ← true
17:         End If
18:     End while
19: End For
20: End
```

to sort the set E using the quick-sort algorithm. After that, we perform the binary search according to one objective only to conclude the correct rank of a solution. This operation requires $log(|E|)$ computations. So thus, we need $O(|E|log(|E|)) + O(log(|E|)) \sim O(|E|log(|E|))$ to determine the rank of such solution u belongs to $\{P \setminus E\}$. The worst case requires $(M-2)[|E|log(|E|)]$ to sort the population according to the last $M-2$ objectives. Since the population members are already sorted according f_1 and f_2 in the bi-objective sorting. Moreover, we need in this case $Mlog(|E|)$ computations to perform the binary search. We deduce then a complexity of $(M-2)[|E|log(|E|)]$ needed to correct the rank of such solution u in this situation. Now, for the average case, we necessitate $O(((NV-2)|E|log(|E|)) + (NVlog(|E|))) \sim O((NV-2)|E|log(|E|))$ time computations where NV belongs to $[1..M]$. In general, the overall complexity of the rank correction procedure for the $\{P \setminus E\}$ solutions is as follows.

1) A quasi-linear time complexity of $O((N-|E|)[|E|log(|E|)])$ for the best case.
2) A quasi-linear time complexity of $O((N-|E|)[(M-2)|E|log(|E|)])$ for the worst case.
3) A quasi-linear time complexity of $O((N-|E|)[(NV-2)|E|log(|E|)])$ for the average case.

In summary, we can say that the rank correction procedure presents a quasi-linear time complexity in the three cases. As well, it is important to note that the complexity of this procedure decreases following each front identification since the population is truncated.

We study now in this paragraph the computational cost of our Q-NDSA for the M-objective case from a time viewpoint. It is worth noting that we use in the following the worst case complexity of the rank correction method to calculate the overall complexity of Q-NDSA.

1) The best case needs $O(Nlog(N))$ for the bi-objectives sorting and $O((N-|E|)(M-2)[|E|log(|E|)])$ for the rank correction. Thus the overall complexity in this situation $O(Nlog(N)) + O((N-|E|)(M-2)[|E|log(|E|)]) \sim O(Nlog(N))$.
2) The worst case needs $O(N^2) + O((N-|E|)(M-2)[|E|log(|E|)]) \sim O(N^2)$.
3) The average case requires as well $O(Nlog(N))$ for the bi-objective sorting and $O((N-|E|)(M-2)[|E|log(|E|)])$ to correct the rank of $\{P \setminus E\}$. The overall complexity in this case is $O(NlogN) + O((N-|E|)(M-2)[|E|log(|E|)]) \sim O(Nlog(N))$.

C. NCRO Computational Complexity

In this section, we compute the NCRO overall complexity for the best, average, and worst cases. The NCRO basic operations are: 1) the Q-NDSA sorting; 2) the crowding distance assignment; and 3) the sorting according fitness value. Firstly, the Q-NDSA has as average and best case complexities $O(Nlog(N))$ and as worst case complexity $O(N^2)$ computations. The crowding distance assignment requires $O(MNlog(N))$ computations for the best, average and worst case. And the sorting according to fitness value takes $O(Nlog(N))$ computations for the three situations. In summary, the overall complexity of the NCRO is $MNlog(N)$ for the best and average case which is governed by crowding distance computation process. However, for the worst case, NCRO presents a quadratic time complexity of $O(N^2)$.

IV. Experimental Study

In this section, we validate the performance of NCRO algorithm. Our experiments can be divided into two parts. The first one is devoted to compare NCRO against four well-cited MOEAs having different fitness assignment schemes. The algorithms are as follows.

1) NSGA-II which is based on Pareto-based ranking.
2) IBEA which uses indicator-based selection.
3) AGE which operates with a formal notion of approximation improved during the iterative process.
4) MOEA/D which decomposes the MOP into N subproblems (we use the Tchebycheff version of MOEA/D).

The second one is dedicated to CPU time analysis in order to assess the efficiency of our approach from a computational time viewpoint. We note that all algorithms are coded in Java programming language and all simulations are performed on the same machine (Intel Core i7-4500 CPU 1.8 GHz, 8 GB RAM).

A. Benchmark Problems

In this subsection, we describe the different benchmark problems used in our experimental study. In fact, we use commonly used test problems within the EMO community that allow assessing the performance of any multi-objective metaheuristic with respect to different kinds of difficulties. We choose to use then the first seven test problems of DTLZ suite (DTLZ1-7) [39] in addition to the WFG5 problem [40]. Moreover we use two variants of DTLZ1 and DTLZ2 called scaled DTLZ1 (S-DTLZ1) and scaled-DTLZ2 (S-DTLZ2) [41]. These two latter present an

additional difficulty consisting in using different scales for the objective functions. We recall that DTLZ1 has a linear Pareto front and is multimodal. DTLZ2 optimal front lies in the first quadrant of the unit sphere in a three-objective plot. DTLZ3 has several local Pareto fronts that are parallel to the optimal front. For DTLZ4 and DTLZ5, the optimal solutions are nonuniformly distributed along the optimal front. For DTLZ6, the front is a curve and the solution density gets thinner toward the Pareto front. As well we choose the WFG5 [40] test problem from WFG suite to cover the deceptiveness difficulty.

B. Performance Measure

In our experimental study we use the hypervolume indicator (HV) which corresponds to the proportion of the objective space that is dominated by the Pareto front approximation returned by the algorithm and delimited by a reference point. Larger values for this metric mean better performance. This metric assesses both convergence and diversity. The used reference point here corresponds to the nadir point obtained over all Pareto front approximations of all runs [42].

TABLE I
HV VALUES OF NCRO, NSGA-II, IBEA, MOEA/D, AND AGE FOR THE BI-OBJECTIVE CASE. THE SYMBOL "+" MEANS THAT H_0 IS REJECTED WHILE THE SYMBOL "−" MEANS THE OPPOSITE. THE BEST VALUES ARE HIGHLIGHTED IN BOLD

Problem	NCRO	NSGA-II	IBEA	MOEA/D	AGE
DTLZ1	0.4907 (+ + + +)	0.4874 (+ + + +)	0.2485 (+ + + +)	**0.4931 (+ + + -)**	0,3749 (+ + + -)
DTLZ2	**0.2149 (+ + + +)**	0.2122 (+ + + +)	0.2133 (+ + + -)	0.2136 (+ + + -)	0,2147 (+ + - -)
DTLZ3	**0.1158 (+ + + +)**	0.0951 (+ + + +)	0.0274 (+ + + -)	0.1015 (+ + + -)	0.0101 (+ + - -)
DTLZ4	0.2099 (+ + + +)	0.2091 (+ + + +)	0.2094 (+ + + -)	0,2094 (+ + + -)	**0,2108 (+ + - -)**
DTLZ5	**0.2146 (+ + + +)**	0.2139 (+ + + +)	0.2132 (+ + + -)	0.2146 (+ + + -)	0,2147 (+ + - -)
DTLZ6	0.0755 (+ + + +)	0.1655 (+ - + +)	0.1623 (+ - + -)	**0.2149 (+ + + -)**	0.006 (+ + - -)
DTLZ7	0.3145 (+ - + -)	0.3319 (+ + + +)	0.3299 (- + - -)	0.3251 (+ + - -)	**0,3307 (- + - -)**
SDTLZ1	**0.4890 (+ + + +)**	0.4878 (+ + + +)	0.2430 (+ + + -)	0.4720 (+ + + -)	0,3148 (+ + - -)
SDTLZ2	**0.2148 (+ + + +)**	0.2140 (+ + + +)	0.2128 (+ + + -)	0.2032 (+ + + -)	0,2142 (+ + - -)
WFG5	0.1953 (+ + + +)	0.1948 (+ + - +)	**0.1969 (+ + + -)**	0.1946 (+ - + -)	0,1965 (+ + - -)

C. Parameter Settings

We detail the parameter setting for each of the five algorithms under comparison. In order to ensure fairness of comparisons, we use the same number of function evaluations (FEs) as a termination criterion. In fact, we use 25 000 FEs for the bi-objective case and 50 000 FEs for the tri-objective one. The other common used parameters setting are as follows: population size, crossover probability, mutation probability, simulated binary crossover (SBX) distribution index and polynomial mutation index which are set to 100 for bi-objective case and 250 for the tri-objective one, 0.9, 1/n, 20, 20, respectively. Regarding the specific parameters, the MOEA/D neighborhood size is set to 20 and the fitness scaling factor for IBEA is set to 0.05. The NCRO specific parameters are InitialKE, KELoss Rate, MoleColl, DecThres, and SynThresare which are set to 10 000, 0.6, 0.7, 15, 10, respectively. It is important to note that both decomposition and synthesis operations share similarities with the crossover and mutation operations. Therefore, SBX crossover and polynomial mutation can be implemented easily in decomposition and synthesis operations [22]. The crossover operator is implemented in synthesis to combine solutions into one. While the polynomial mutation operator can be applied to a molecule twice to produce two different molecules. We mention here that we have used the trial-and-error method for tuning parameter values [25].

D. Comparative Results

We choose to use the Wilcoxon rank sum test in a pairwise fashion [16]. Our choice is justified by the fact that the Wilcoxon test is nonparametric since it works on the values' ranks and not on the values themselves. It allows to verify whether the results are statistically different or not between samples. In fact, the minimum sample size considered acceptable for this test needs to be stipulated. There is no established agreement about this specification [16]. Therefore, statisticians have studied the minimum sample size needed and they suggested that a sample size of 30 is the minimum size required to ensure a certain power of the statistical test [46]. In our case, we use 31 runs to facilitate the median extraction. For each couple (algorithm, problem), we perform 31 runs with different random seeds (i.e., 31 different randomly generated populations). Once the results are obtained, we use the MATLAB ranksum function in order to compute the p-value of the following hypothesis: 1) H_0: median (Algorithm 1) = median (Algorithm 2) and 2) H_1: median (Algorithm 1) $\neq$ median (Algorithm 2) for a confidence level of 95%, if the p-value is found to be less or equal than 0.05, we reject H_0 and we accept H_1. In this way, we can say that the medians of the two algorithms are different from each other and that one algorithm outperforms the other viewpoint the used metric. However, if the p-value is found to be greater than 0.05, then we accept H_0 and we cannot say that one algorithm is better than the other nor the opposite.

The simulation results of five algorithms for each test problem with respect to HV metric on the bi-objective case are summarized in Table I. We observe that NCRO presents a good performance on the majority of test problems. It outperforms NSGA-II, IBEA, MOEA/D, and AGE on DTLZ2, DTLZ3, DTLZ5, SDTLZ1, and SDTLZ2 which can be explained by the driving force of CRO in manipulating both diversity and convergence operators under the search space. However, NCRO gives a bad compromise between convergence and diversity on DTLZ6 and DTLZ7 problems. As well, we note that IBEA obtains a worse HV contribution in the majority of test problems. Nevertheless, it was able to surpass its competitors on WFG5 problem. Note that MOEA/D and AGE algorithms can maintain a good HV value on both set of problems DTLZ1, DTLZ6 and DTLZ4, DTLZ7, respectively. In general, we deduce that NCRO can maintain a good diversity of solutions on the Pareto-optimal front and can converge on to the

TABLE II
HV VALUES OF NCRO, NSGA-II, IBEA, MOEA/D, AND AGE FOR THE TRI-OBJECTIVE CASE. THE SYMBOL "+" MEANS THAT H_0 IS REJECTED WHILE THE SYMBOL "−" MEANS THE OPPOSITE. THE BEST VALUES ARE HIGHLIGHTED IN BOLD

Problem	NCRO	NSGA-II	IBEA	MOEA/D	AGE
DTLZ1	**0.7913** (+ + + +)	0.7903 (+ + + +)	0.3630 (+ + + -)	0.7757 (+ + + -)	0.7905 (+ + - -)
DTLZ2	0.4189 (+ + + +)	0.4109 (+ + + +)	0.4280 (+ + + -)	0.4086 (+ + + -)	**0.4387** (+ + - -)
DTLZ3	**0.4021** (- + + +)	0.4010 (**- + + +**)	0.0143 (+ + + -)	0.3597 (+ + + -)	0.0001 (+ + - -)
DTLZ4	0.4153 (+ + + +)	0.4101 (+ + + +)	**0.4211** (**+ + + +**)	0.4058 (+ + + -)	0.4160 (+ + + -)
DTLZ5	**0.0953** (+ + + +)	0.0950 (+ + + +)	0.0942 (+ + + -)	0.0940 (+ + + -)	0.0951 (+ + - -)
DTLZ6	0.0271 (+ + +)	**0.0960** (+ + +)	0.0925 (+ + +)	0.0953 (**+ + +**)	0.0002 (+ + - -)
DTLZ7	0.3015 (+ + + +)	0.3078 (+ + + +)	0.2924 (+ + + -)	0.2256 (+ + + -)	**0,3094** (**+ + - -**)
SDTLZ1	**0.7903** (**- + + +**)	0.7901 (- + + +)	0.3711 (+ + + -)	0.5885 (+ + + -)	0,7819 (+ + - -)
SDTLZ2	**0.4290** (+ + + +)	0.4110 (+ + + +)	0.4281 (+ + + -)	0.2651 (+ + + -)	0,4179 (+ + - -)
WFG5	0.3802 (+ + + +)	0.3753 (+ + + +)	0.3906 (+ + + -)	0.3589 (+ + + -)	**0,4021** (+ + - -)

Pareto optimal front. Table II shows the obtained HV values on tri-objective case. NCRO presents the better results on DTLZ1, DTLZ3, DTLZ5, SDTLZ1 and SDTLZ2. This observation means that NCRO works well in terms of convergence and diversity for problems involving nonuniformly and linear Pareto front, several local-optima and different scaled objectives. However, for the DTLZ6 problem, similarly to the bi-objective case, NCRO presents a bad compromise between convergence and diversity. We can conclude that NCRO cannot give a good result with problems possessing curve front. IBEA is able to give a good HV contribution on DTLZ4 and WGF5. The worst performance is always observed with MOEA/D on SDTLZ1 and SDTLZ2 problems. For DTLZ7, AGE performs well viewpoint convergence and diversity compared to IBEA, MOEA/D, and NCRO.

The Wilcoxon rank sum test allows verifying whether the results are statistically different or not. However, it does not give any idea about the difference magnitude. The effect size could be computed by using the Cohen's d statistic [46]. The effect size is considered: 1) small if $0.2 \leq d < 0.5$; 2) medium if $0.5 \leq d < 0.8$; or 3) large if $d \geq 0.8$. Based on effect size analysis, we obtained the following.

1) On Scaled DTLZ problem, we have a large difference between NCRO and MOEA/D and medium difference between NCRO and the others.
2) On DTLZ3, we have a large difference between NCRO and AGE, as well between NCRO and IBEA and medium difference between NCRO and the others.
3) On DTLZ1, we have a large difference between NCRO and AGE also we note a large difference between NCRO and IBEA; a small difference is observed between NCRO and the others.
4) Considering DTLZ2, DTLZ4, and DTLZ5, we have a small difference between NCRO and the four competitors.

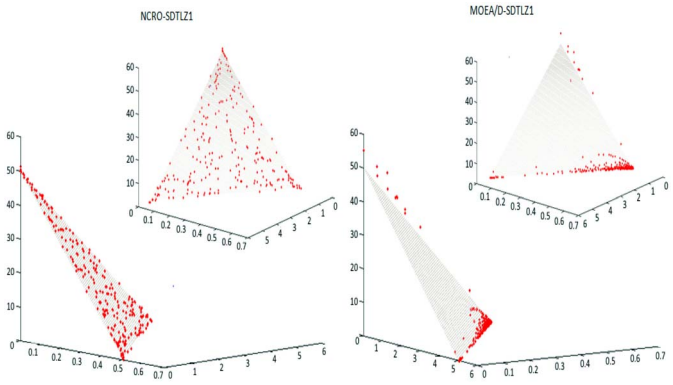

Fig. 6. Nondominated solutions with MOEA/D and NCRO on SDTLZ1.

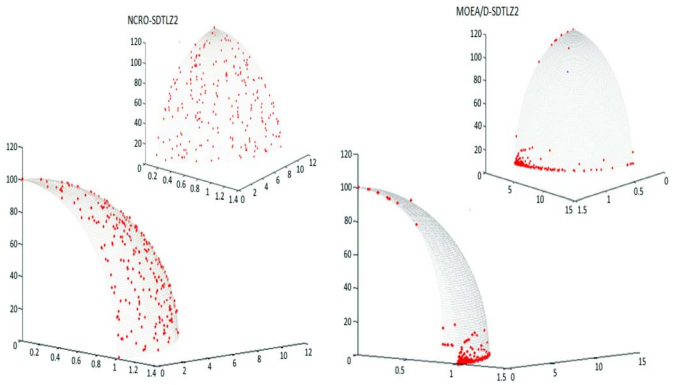

Fig. 7. Nondominated solutions with MOEA/D and NCRO on SDTLZ2.

5) On DTLZ6, we obtain a large effective size between NCRO and other algorithms.
6) On DTLZ7, we have a large difference between NCRO and AGE and small difference between NCRO and the others.
7) Finally, considering WFG5, we have a large difference between NCRO and MOEA/D and a medium difference between NCRO and the others.

The above DTLZ problems have one common aspect—all objectives are equally scaled for the Pareto-optimal solutions. Thus, to investigate whether the performance of NCRO method is due to this property of the DTLZ, we assess its performance on two modified versions of DTLZ1 and DTLZ2 with differently scaled objective values and we compare then the results with the MOEA/D algorithm. We show the plots of the obtained nondominated solutions, in Figs. 6 and 7, for NCRO and MOEA/D. We can observe that MOEA/D suffers from scaling of objectives. Similar results are observed for the scaled SDTLZ2 problem as well. Clearly, we can deduce from both statistical results and from Figs. 6 and 7 that NCRO outperforms MOEA/D algorithm on these two test problems which means that NCRO can give a good performance regarding scaled problems. It is worth noting that MOEA/D has demonstrated similar results on scaled problems when compared against NSGA-III [5].

TABLE III
NUMBER OF COMPARISONS AND CPU TIME CALCULATED BY THE Q-NDS ALGORITHM AND DEB'S ALGORITHM. THE CPU VALUES ARE WRITTEN BETWEEN PARENTHESIS

Popula-tion size	Number of objectives					
	2		3		4	
	Q-NDSA	Deb's algo-rithm	Q-NDSA	Deb's algo-rithm	Q-NDSA	Deb's algo-rithm
100	1398 (2ms)	12995 (14ms)	2418 (19ms)	15908 (22ms)	2449 (15ms)	16881 (29ms)
200	3559 (4ms)	51913 (39ms)	5526 (26ms)	58693 (50ms)	5628 (27ms)	64695 (70ms)
300	6791 (7ms)	117890 (72ms)	9561 (38ms)	130013 (114ms)	11246 (55ms)	138927 (123ms)

E. CPU Time Comparison

In addition to the empirical simulation presented to assess the performance of different MOEAs regarding HV metric, we are also interested in CPU time analysis. At the beginning of this paper, we mentioned that NCRO runs quickly regarding NSGA-II, since it uses the Q-NDSA with quasi-linear complexity. To further emphasize this theoretical result, we first compare the new Q-NDSA algorithm with Deb *et al.*'s [9] algorithm to see how the former reduces the number of comparison among solutions as well as the CPU time, for different numbers of objectives and population size. After that, we compare the total CPU time of NCRO regarding NSGA-II, IBEA, MOEA/D, and AGE. We implemented the Q-NDSA and Deb *et al.*'s [9] algorithm using Java language, so, it was easy to use the same populations for comparing both algorithms. We compare the new algorithm using three populations on test problems with two to four objectives, respectively. The results are given in Table III. We can see that Q-NDSA reduces the number of comparisons among solutions as well as the CPU time, regarding Deb *et al.*'s [9] algorithm. The reason is that Q-NDSA presents a new technique able to generate the same results as Deb *et al.*'s [9] algorithm with fewer comparison number. To get the true performance of the NCRO algorithm we need also to evaluate the total processing time. Fig. 8 reports CPU times for each MOEA used in this experimental study. We choose here for brevity, only two test problems DTLZ2 and WFG5. Each one of them is characterized by different difficulties. We use then the FEs as a termination criterion in which 25 000 and 50 000 are the number of FEs used in the bi-objectives and tri-objective case, respectively. We observe that NCRO represents a CPU times less than NSGA-II, AGE, and IBEA. While, compared to MOEA/D, NCRO takes a little more CPU times. For the tri-objective case, we observe in Fig. 9 the same comparison results between NCRO, NSGA-II, AGE, and IBEA. However, for MOEA/D algorithm we notice that this algorithm becomes more efficient than NCRO. This observation can be explained as follows.

1) NCRO uses a new quick nondominated sorting with quasi-linear complexity which makes it quickly regarding such Pareto evaluation-based algorithm.
2) The IBEA algorithm uses the hyper-volume indicator in the selection process which consumes a lot of time in its computation.

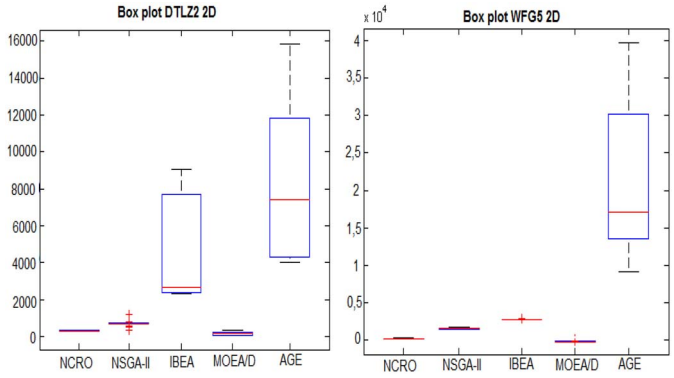

Fig. 8. CPU time boxplots for NCRO, NSGA-II, IBEA, MOEA/D, and AGE on DTLZ2 and WFG5 test problems for the bi-objective case.

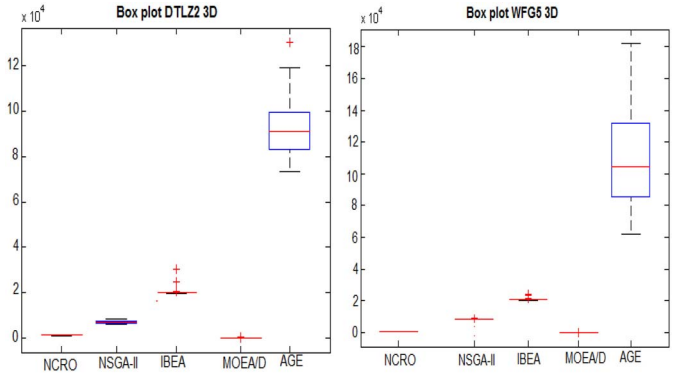

Fig. 9. CPU time boxplots for NCRO, NSGA-II, IBEA, MOEA/D, and AGE on DTLZ2 and WFG5 test problems for the tri-objective case.

3) AGE algorithm requires a high computational cost with a large size population, which makes it very slow [45].
4) MOEA/D is a decomposition based algorithm. It consumes $O(MNT)$ computational complexity, where N corresponds to the population size, M is the number of objectives and T is the number of the weight vectors in the neighborhood of each weight vector.

Thus, using the same population size, the ratio between NCRO and MOEA/D computational complexities at each generation is

$$\frac{O(MNT)}{O(MNlogN)} = \frac{O(T)}{O(logN)}. \tag{3}$$

Since T is smaller than $log\,(N)$, MOEA/D requires lower computational complexity than NCRO which may explain the obtained comparison results between NCRO and MOEA/D. In summary, we say that the CPU times analysis shows clearly that NCRO outperforms NSGA-II, IBEA, and AGE in terms of CPU time which can make it an efficient Pareto dominance based-MOEA.

F. Discussion

This section is devoted to discuss the obtained results with an attempt to explain the NCRO behavior. Indeed, NCRO seems competitive when compared to several state of the art MOEA from HV viewpoint. This observation could be

TABLE IV
HV VALUE OF NCRO ON DIFFERENT TEST PROBLEMS ACCORDING TO THE TEN PARAMETER TUNING COMBINATIONS. THE BEST VALUES ARE HIGHLIGHTED IN BOLD

	InitialKE	KELossRate	MoleColl	DecThres	SynThres	DTLZ1	DTLZ2	DTLZ3	DTLZ4	DTLZ5	DTLZ6	DTLZ7	SDTLZ1	SDTLZ2	WFG5
1	1000	0.2	0.2	15	10	0.716365	0.406779	0.392101	0.413347	0.095049	0.00171	0.274222	0.776806	0.409174	0.370453
2	850	0.4	0.5	50	20	0.708789	0.402545	0.372101	0.402495	0.085214	0.000102	0.259919	0.739267	0.408666	0.367415
3	10000	0.6	0.7	200	100	0.784741	0.421965	0.366210	0.413690	0,095215	0.000267	0.279643	0.774972	0.416575	0.378422
4	1000000	0.8	0.9	15	10	0,782765	0.412024	0.282806	0.411942	0.095198	0.00270	0.284846	0.786310	0.415756	0.377656
5	1000	0.4	0.7	50	10	0,786264	0.417288	0.322806	0.413686	0.095178	0.00264	0.27499	0.743387	0.416352	0.368122
6	850	0.6	0.9	200	100	0,754485	0,404216	0,352280	0.408718	0.094205	0.00256	0.286929	0.754352	0.408172	0.366727
7	**10000**	**0.6**	**0.7**	**15**	**10**	**0.791302**	**0.418944**	**0.402101**	**0.416056**	**0.095374**	**0.00271**	**0.301546**	**0.784508**	**0.427204**	**0.384374**
8	10000	0.4	0.2	50	20	0.780658	0.421747	0.392806	0.415914	0.095179	0.00242	0.283218	0.780574	0.416595	0.377013
9	850	0.2	0.5	15	100	0.712569	0.407976	0.339828	0.411187	0.095108	0.00233	0.261214	0.747212	0.411022	0.376912
10	1000	0.8	0.2	50	10	0.768129	0.413885	0.400280	0.407381	0.095270	0.00221	0.279747	0.784496	0.419432	0.377284

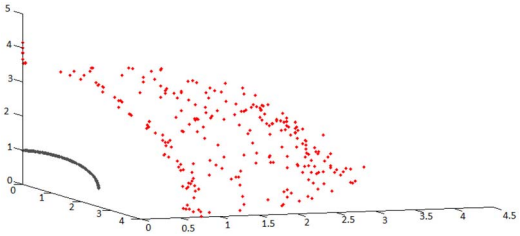

Fig. 10. Nondominated solutions with NCRO on DTLZ6. The gray line is the true Pareto front.

explained by the fact that NCRO guides the population individuals by means of chemical reactions. Among these latter, we find on-wall ineffective collision and interineffective one which could be considered as local search operators. The first one applies local search in a single neighborhood while the second one makes a search in two local neighborhoods. These kind of operators emphasizes the convergence of NCRO. We have also noted a good diversity performance of NCRO for several problems' difficulties. The main reasons may be: 1) the ability of NCRO in well-controlling the trade-off between intensification and diversification based on its four operators and 2) the control of search directions thanks to the use of kinetic energy. NCRO could avoid visiting nonpromising regions in the search space thanks to the PE management rules, thereby improving convergence and diversity. Therefore, NCRO gives a competitive results regarding all the four competitors on DTLZ3, DTLZ5, and the two scaled DTLZ problems. It seems to be that NCRO has a good performance with problems involving several local Pareto fronts, problems characterized by nonuniformly distributed optimal solutions and with scaled ones. Indeed, we remark that NCRO outperforms NSGA-II in all test problems. Nevertheless, it does not present a good performance for few instances of test problems like DTLZ7 which is characterized by disconnectedness. As well, for the DTLZ6, NCRO presents a bad performance. This problem is characterized by the fact that solution density gets thinner toward the Pareto front [40]. The difficulty of this problem mainly reflects in poor convergence to the optimal front since in this problem the Pareto optimal set lies at the bounds of the decision space [40]. This problem imposes higher convergence difficulties for most of the studied approaches. On the one hand, NCRO was the method which achieved relatively low values on this problem. On the other hand, AGE seems to be the most affected method by the DTLZ6's difficulties. This observation is explained by the fact that NCRO was unable to detect the boundary points which lead to a bad performance. Fig. 10 presents a plot describing the NCRO performance on DTLZ6 problem. The lack of convergence to the true front in this problem causes NCRO to find a dominated surface, whereas the true Pareto-optimal front is a curve. In summary and based on the obtained experiment results, we can conclude that the proposed algorithm has been successful in providing a good balance between convergence and diversity. From an efficiency viewpoint, we have demonstrated theoretically and experimentally that NCRO requires less CPU times than all algorithms except for MOEA/D (which is based on decomposition).

Since the setting of parameter values of an optimization algorithm can have a profound impact on its performance, we would discuss here the parameter tuning of our NCRO procedure. In fact, the NCRO parameter values are deduced from our implementation using the trial-and-error method. However, these parameter values are problem dependent. To maximize the performance of NCRO for a particular problem, we perform some parameter tunings to determine a good combination of these parameters. Therefore, to clarify this issue, we tested the performance of NCRO on different parameter combinations. For each parameter, we choose a value set selected from [23], [47], and [48] and we presented then the HV contribution for each parameter combination (Table IV). The set value chosen for every parameter are presented as follows.

1) *InitialKE:* 1000, 850, 10 000, 10 00 000.
2) *KELossRate:* 0.2, 0.4, 0.6, 0.8.
3) *MoleColl:* 0.2, 0.5, 0.7, 0.9.
4) *DecThres:* 15, 50, 200.
5) *SynThres:* 10, 20, 100.

We deduced from Table IV, the parameter sensitivities for the search performance of NCRO. In fact, each parameter influences particularly the performance behavior of our approach. Begin with InitKE, this parameter controls the total energy of the system. Therefore, a higher value increases the convergence rate of NCRO, but this is at the expense of getting stuck in a local minimum. Thus, the best InitKE value deduced from our implementation is set to 10 000. Concerning the MoleColl parameter, we remark from Table IV that a low value of this parameter leads to a

weak HV contribution compared to a high value of MoleColl. Therefore, this parameter controls the choice of collision types (unimolecular collision or intermolecular collision). A higher value means a higher chance to execute intermolecular collisions which enhance the exploration of a new region of the solution space and visiting several neighborhoods. The KELossRate parameter is a system parameter which limits the maximum percentage of kinetic energy lost at a time. The best value leading to a good HV contribution is set to 0.6 for the majority of test problem. The DecThres and SynThres control the Synthesis and decomposition occurrence. In fact a higher value of Synthesis leads to a higher rate of diversification, thereby destroying the convergence rate of the algorithm. Thus, we fixed this threshold to 10. Finally, we set the DecThres to 15 to control the number of molecule moves for certain time. In fact, a higher value of this parameter can lead to a local minimum since the molecule intensifies the search in a particular region for a long time without executing a diversification operator (i.e., decomposition). Consequently, as we described, NCRO requires the setting of several parameters. Such task is performed in this paper by trial and error. Consequently, it would be interesting to design an adaptive parameter control strategy for NCRO in future work.

V. Conclusion

In this paper, we have suggested, for the first time, a multiobjective CRO algorithm using a newly proposed nondominated sorting algorithm (Q-NDSA). The experimental results revealed that NCRO is very competitive with respect to the considered algorithms. We would like to discuss in this section the possible threats to validity that could be classified into three categories: 1) the construct validity; 2) the internal validity; and 3) the external one. In our experimental study, a threat to construct validity arises because, although we considered the well-used HV quality metric, we did not use other quality metrics. In future work, we plan to use additional quality metrics to evaluate the performance of our approach. Major internal threats involve the stochastic behavior of algorithms. In fact, the parameter tuning of our approach are settled by the trial-and-error method. Although, the obtained results are promising, it would be a challenging perspective to design an adaptive parameter control strategy for NCRO. As well, it is interesting to investigate in future work the NCRO behavior in the decision space, e.g., presenting the HV transition while visualizing the solution search. External validity refers to the generalizability of our results. Our experiments are based on a variety of benchmark problems presenting various challenging difficulties. However, we cannot assert that our results can be generalized to real word problems (see [49]) since the latter usually involve several types of constraints to handle. Thus, another path for future investigation would be the proposal of a constraint handling strategy for NCRO. Finally, recently, there is a growing need for solving dynamic MOPs where objective functions and constraints may vary over time [50] and for incorporating users' preferences in EMO [51]. Thus, it would be challenging to design a preference-based dynamic multiobjective version of NCRO.

Appendix

In the following, we give the basic pseudocode of NCRO.

Algorithm 3 NCRO Pseudocode

Input: *PopSize*, *KELossRate*, *MoleColl*, *InitialKE*, α, β, Number of objectives M, Number of decision variables n
Output: Pareto front approximation P_t
01: **Begin**
02: $t \leftarrow 0$
03: $P_t \leftarrow$ **MakeNewPopulation()** /*Generate randomly the initial population
04: **While** (**NOT** TerminationCriterion) **do**
05: $Q_t \leftarrow$ **CROVariation** (P_t) /*Apply the CRO operators to P_t
06: $Q_t \leftarrow$ **Update** (Q_t) /*Assign *PEvalue* to each individual from Q_t population then apply the energy management laws of CRO*/
07: $P_{t+1} \leftarrow P_t \cup Q_t$
08: $P_{t+1} \leftarrow$ **Q-NDSA** (P_{t+1})
09: $P_{t+1} \leftarrow$ **CrowdingDistanceAssignement** (P_{t+1})
10: $P_{t+1} \leftarrow$ **PotentialEnergyAssignement** (P_{t+1}) /*Compute the PE values for the combined population individuals*/
11: $P_{t+1} \leftarrow$ **BestSolutions** (P_{t+1})
12: $t \leftarrow t+1$
13: **End while**
14: **End**

References

[1] K. Deb, *Black Multi-Objective Optimization Using Evolutionary Algorithms*. New York, NY, USA: Wiley, 2001.

[2] C. A. C. Coello, G. B. Lamont, and D. A. V. Veldhuizen, *Black Evolutionary Algorithms for Solving Multi-Objective Problems*. Berlin, Germany: Springer, 2007.

[3] K. Miettinen, *Nonlinear Multiobjective Optimization*. Dordrecht, The Netherlands: Kluwer Academic, 1999.

[4] C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization," in *Proc. Int. Conf. Genet. Algorithm (ICGA)*, San Francisco, CA, USA, 1993, pp. 416–423.

[5] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point based non-dominated sorting approach, part I: Solving problems with box constraints," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 577–601, Aug. 2014.

[6] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point based non-dominated sorting approach, part II: Solving problems with box constraints," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 602–622, Aug. 2014.

[7] J. Horn, N. Nafpliotis, and D. E. Goldberg, "A niched Pareto genetic algorithm for multiobjective optimization," in *Proc. 1st IEEE Conf. Evol. Comput.*, Orlando, FL, USA, 1994, pp. 82–87.

[8] S. Nidamarthi and K. Deb, "Multiobjective optimization using nondominated sorting in genetic algorithms," *Evol. Comput.*, vol. 2, no. 3, pp. 221–248, 1994.

[9] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[10] E. Ziztler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 257–271, Nov. 1999.

[11] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization," *Evol. Methods Design Optim. Control (EUROGEN)*, Barcelona, Spain, 2001, pp. 95–100.

[12] D. W. Corne, J. D. Knowles, and M. J. Oates, "The Pareto envelope-based selection algorithm for multi-objective optimisation," in *Proc. Parallel Probl. Solving Nat. (PPSN VI)*, Paris, France, 2000, pp. 839–848.

[13] E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," in *Proc. Parallel Probl. Solving Nat. (PPSN VIII)*, Birmingham, U.K., 2004, pp. 832–842.

[14] A. Auger, J. Bader, D. Brockhoff, and E. Zitzler, "Hypervolume-based multiobjective optimization: Theoretical foundations and practical implications," *Theor. Comput. Sci.*, vol. 425, no. 1, pp. 75–103, 2012.

[15] D. H. Phan and J. Suzuki, "R2-IBEA: R2 indicator based evolutionary algorithm for multiobjective optimization," in *Proc. IEEE Congr. Evol. Comput.*, Cancun, Mexico, 2013, pp. 1836–1845.

[16] J. Derrac, S. García, and D. Molina, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, 2011.
[17] N. Beume, B. Naujoks, and M. Emmerich, "SMS-EMOA: Multiobjective selection based on dominated hypervolume," *Eur. J. Oper. Res.*, vol. 181, no. 3, pp. 1653–1669, 2007.
[18] H. Trautmann, T. Wagner, and D. Brockhoff, "R2-EMOA: Focused multiobjective search using R2-indicator-based selection," in *Proc. Learn. Intell. Optim.*, Catania, Italy, 2013, pp. 70–74.
[19] H. Ishibuchi and T. Murata, "A multi-objective genetic local search algorithm and its application to flowshop scheduling," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 28, no. 3, pp. 392–403, Aug. 1998.
[20] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.
[21] D. K. Saxena, J. A. Duro, A. Tiwari, K. Deb, and Q. Zhang, "Objective reduction in many-objective optimization: Linear and nonlinear algorithms," *IEEE Trans. Evol. Comput.*, vol. 17, no. 1, pp. 77–99, Feb. 2013.
[22] A. Y. S. Lam and V. O. K. Li, "Chemical reaction-inspired metaheuristic for optimization," *IEEE Trans. Evol. Comput.*, vol. 14, no. 3, pp. 381–399, Jun. 2010.
[23] A. Y. S. Lam, V. O. K. Li, and J. J. Q. Yu, "Real-coded chemical reaction optimization," *IEEE Trans. Evol. Comput.*, vol. 16, no. 3, pp. 339–353, Jun. 2012.
[24] Y. Sun, A. Y. S. Lam, V. O. K. Li, J. Xu, and J. J. Q. Yu, "Chemical reaction optimization for the optimal power flow problem," in *Proc. IEEE Congr. Evol. Comput.*, Brisbane, QLD, Australia, 2012, pp. 1–8.
[25] A. E. Eiben and K. S. Selmar, "Parameter tuning for configuring and analyzing evolutionary algorithms," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 19–31, 2011.
[26] C. L. Hwang and A. S. M. Masud, *Introduction in Multiple Objective Decision Making: Theory and Applications* (Lecture Notes in Economics and Mathematical Systems). New York, NY, USA: Springer, 1979, pp. 1–7.
[27] L. Ke, Q. Zhang, and R. Battiti, "MOEA/D-ACO: A multiobjective evolutionary algorithm using decomposition and ant colony," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 1845–1859, Dec. 2013.
[28] A. Zhou *et al.*, "Multiobjective evolutionary algorithms: A survey of the state of the art," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 32–49, 2011.
[29] L. Ke, Q. Zhang, and R. Battiti, "Hybridization of decomposition and local search for multiobjective optimization," *IEEE Trans. Cybern.*, vol. 44, no. 10, pp. 1808–1820, Oct. 2014.
[30] J. J. Q. Yu, A. Y. S. Lam, and V. O. K. Li, "Real-coded chemical reaction optimization with different perturbation functions," in *Proc. IEEE Congr. Evol. Comput.*, Brisbane, QLD, Australia, 2012, pp. 1–8.
[31] J. Xu, A. Y. S. Lam, and V. O. K. Li, "Parallel chemical reaction optimization for the quadratic assignment problem," in *Proc. Int. Conf. Genet. Evol. Methods*, Las Vegas, NV, USA, 2010, pp. 125–131.
[32] B. Pan, A. Y. S. Lam, and V. O. K. Li, "Network coding optimization based on chemical reaction optimization," in *Proc. IEEE Global Commun. Conf.*, Houston, TX, USA, 2011, pp. 1–5.
[33] J. J. Q. Yu, A. Y. S. Lam, and V. O. K. Li, "Evolutionary artificial neural network based on chemical reaction optimization," in *Proc. IEEE Congr. Evol. Comput.*, New Orleans, LA, USA, 2011, pp. 2083–2090.
[34] M. T. Jensen, "Reducing the run-time complexity of multiobjective EAs: The NSGA-II and other algorithms," *IEEE Trans. Evol. Comput.*, vol. 7, no. 5, pp. 503–515, Oct. 2003.
[35] M. A. Yukish, "Algorithms to identify Pareto fronts in multi-dimensional data sets," Ph.D. dissertation, The Graduate School, College of Eng., Pennsylvania State Univ., State College, PA, USA, 2004.
[36] H. T. Kung, F. Luccio, and F. P. Preparata, "On finding the maxima of a set of vectors," *J. ACM*, vol. 22, no. 4, pp. 469–476, 1975.
[37] H. Fang, Q. Wang, Y.-C. Tu, and M. F. Horstemeyer, "An efficient non-dominated sorting method for evolutionary algorithms," *Evol. Comput.*, vol. 16, no. 3, pp. 355–384, 2008.
[38] K. McClymont and E. Keedwell, "Deductive sort and climbing sort: New methods for non-dominated sorting," *Evol. Comput.*, vol. 20, no. 1, pp. 1–26, 2012.
[39] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable multi-objective optimization test problems," in *Proc. IEEE Congr. Evol. Comput.*, Honolulu, HI, USA, 2002, pp. 825–830.
[40] S. Huband, P. Hingston, L. Barone, and L. While, "A review of multiobjective test problems and a scalable test problem toolkit," *IEEE Trans. Evol. Comput.*, vol. 10, no. 5, pp. 477–506, Oct. 2006.
[41] K. Deb and H. Jain, "Handling many-objective problems using an improved NSGA-II procedure," in *Proc. IEEE Congr. Evol. Comput.*, Brisbane, QLD, Australia, 2012, pp. 1–8.
[42] A. Auger, J. Bader, D. Brockhoff, and E. Ziztler, "Theory of the hypervolume indicator: Optimal-distributions and the choice of the reference point," in *Proc. 10th ACM SIGEVO Workshop Found. Genet. Algorithms*, Orlando, FL, USA, 2009, pp. 87–102.
[43] K. Bringmann, T. Friedrich, F. Neumann, and M. Wagner, "Approximation guided evolutionary multi-objective optimization," in *Proc. 21st Int. Joint Conf. Artif. Intell. (IJCAI)*, Barcelona, Spain, 2011, pp. 1198–1203.
[44] B. Alatas, "ACROA: Artificial chemical reaction optimization algorithm for global optimization," *Expert Syst. Appl.*, vol. 38, no. 10, pp. 13170–13180, 2011.
[45] B. Alatas, "A novel chemistry based meta-heuristic optimization method for mining of classification rules," *Expert Syst. Appl.*, vol. 39, no. 12, pp. 11080–11088, 2012.
[46] J. Cohen, *Statistical Power Analysis for the Behavioral Sciences*. New York, NY, USA: Psychology Press, 1988.
[47] J. Xu, A. Y. S. Lam, and V. O. K. Li, "Chemical reaction optimization for the grid scheduling problem," in *Proc. IEEE Int. Conf. Commun.*, Cape Town, South Africa, 2010, pp. 1–5.
[48] J. Xu, A. Y. S. Lam, and V. O. K. Li, "Short adjacent repeat identification based on chemical reaction optimization," in *Proc. IEEE Congr. Evol. Comput.*, Brisbane, QLD, Australia, 2012, pp. 1–8.
[49] J. Sun, Q. Zhang, and X. Yao, "Meta-heuristic combining prior online and offline information for the quadratic assignment problem," *IEEE Trans. Cybern.*, vol. 44, no. 3, pp. 429–444, Mar. 2014.
[50] A. Zhou, Y. Jin, and Q. Zhang, "A population prediction strategy for evolutionary dynamic multiobjective optimization," *IEEE Trans. Cybern.*, vol. 44, no. 1, pp. 40–53, Jan. 2014.
[51] S. Bechikh, L. Ben Said, and K. Ghédira, "Negotiating decision makers' reference points for group preference-based evolutionary multi-objective optimization," in *Proc. 11th IEEE Int. Conf. Hybrid Intell. Syst.*, Melacca, Malaysia, 2011, pp. 377–382.

Slim Bechikh received the B.Sc., M.Sc., and Ph.D. degrees in computer science from the University of Tunis, Tunis, Tunisia, in 2006, 2008, and 2013, respectively.

He is currently an Assistant Professor of Computer Science with the University of Carthage, Tunis. His current research interests include multiobjective optimization, evolutionary computation, and their applications.

Abir Chaabani received the B.Sc. and M.Sc. degrees in computer science from the University of Tunis, Tunis, Tunisia, in 2011 and 2013, respectively, where she is currently pursuing the Ph.D. degree from the SOIE Laboratory.

Her current research interests include multiobjective optimization, metaheuristics, bi-level optimization, and their applications.

Lamjed Ben Said received the B.Sc. degree from the University of Tunis, Tunis, Tunisia, in 1998, and the M.Sc. and Ph.D. degrees from the University of Paris VI, Paris, France, in 1999 and 2003, respectively, all in computer science.

He was a Research Fellow with France Telecom (Research and Development), Paris, France, for three years. He is currently an Associate Professor of Computer Science with the University of Tunis, where he is the Head of the SOIE Laboratory and the General Secretary with the Tunisian Association for Artificial Intelligence. His current research interests include multiagent systems, multicriteria decision making, evolutionary computation, and behavioral economics.