A review of the bacterial foraging algorithm in constrained numerical optimization

Betania Hernández-Ocaña
Div. Académica de Informática y Sistemas
Universidad Juárez Autónoma de Tabasco
Tabasco, México
Email:betania.h.o@gmail.com

Efrén Mezura-Montes
Depto. de Inteligencia Artificial
Universidad Veracruzana
Xalapa, Ver., México
Email:emezura@uv.mx

Pilar Pozos-Parra
Div. Académica de Informática y Sistemas
Universidad Juárez Autónoma de Tabasco
Tabasco, México
Email:pilar.pozos@ujat.mx

***Abstract*—A review of the bacterial foraging optimization algorithm used to solve numerical constrained optimization problems is presented in this paper. After an introduction to the algorithm and its main elements, a taxonomy of constraint-handling techniques is presented and adopted to discuss the different approaches based on the algorithm. Aspects related to the most important elements of the algorithm with respect to a constrained search space (e.g., constraint-handling technique, stepsize, tumble-swim operator, reproduction process) are analyzed. Based on the findings of this literature review, some fertile paths of research are presented.**

I. Introduction

Besides evolutionary algorithms (EAs) [1], swarm intelligence algorithms (SIAs) [2] have become popular nowadays to solve complex optimization problems [3]. Regarding SIAs, particle swarm optimization (PSO) was one of the first algorithms to solve numerical optimization problems [4]. In recent years, other SIAs have been proposed to deal with this type of optimization problem, such as the artificial bee colony (ABC) [5] and the bacterial foraging optimization algorithm (BFOA) [6].

Both, EAs and SIAs, grouped as nature-inspired algorithms (NIAs), were originally designed to deal with unconstrained search spaces. Therefore, a constraint-handling technique must be added to them so as to incorporate feasibility information in the search process [7].

The constrained numerical optimization problem (CNOP), without loss of generality, can be defined as to:

minimize $f(\vec{x})$
subject to:
$g_i(\vec{x}) \leq 0$, $i = 1, ..., m$,
$h_j(\vec{x}) = 0$, $j = 1, ..., p$,

where $\vec{x} = [x_1, x_2, \ldots, x_n] \in R^n$, is the solution vector and each decision variable x_i, $i = 1, ..., n$ is bounded by lower and upper limits $L_i \leq x_i \leq U_i$, which define the search space S; m is the number of inequality constraints and p is the number of equality constraints (in both cases, the constraints can be linear or nonlinear). If F denotes the feasible region, then it must be clear that $F \subseteq S$.

Regarding SIAs, the most popular algorithm to solve CNOPs has been PSO [8]. However, other approaches have become attractive in recent years. This is the case of BFOA, which has been adapted to solve CNOPs in different ways as it will be detailed later in this paper. In fact, a recent review of the state-of-the-art in constrained numerical optimization using NIAs showed that the emergence of new search algorithms is a current trend in the area [9].

This work surveys those adaptations made to BFOA to deal with CNOPs, focusing on the type of modification made to the algorithm, advantages and shortcomings. Furthermore, a taxonomy based on the type of constraint-handling mechanism is presented and adopted.

The document is organized as follows, Section II introduces BFOA and its main elements. Section III briefly presents a classification of constraint-handling techniques. Afterwards, Section IV lists those BFOA versions which solve CNOPs, describing their constraint-handling techniques, the modifications made to the algorithms, and their advantages and disadvantages. Section V shows findings obtained from the literature review and, based on them, the future paths of research. Finally, Section VI presents the general conclusions regarding BFOA for CNOPs.

II. BFOA

The bacterial foraging optimization algorithm was proposed by Passino to solve unconstrained numerical optimization problems [6]. BFOA has showed a competitive performance against well-known EAs such as differential evolution (DE) [10], [11], genetic algorithms [12], [13], and also with respect to other SIAs like PSO [14], [15].

BFOA emulates the behavior of the bacterium E.Coli in its search for nutrients in its environment. Each bacterium tries to maximize its obtained energy per unit of time spent on the foraging process while avoiding noxious substances. In fact, bacteria can communicate with others located in their neighborhood. A swarm of biological bacteria S behaves as follows [6]:

1) Bacteria are distributed in the map of nutrients at random.
2) Bacteria locate high-nutrient regions in the map by chemotaxis movements (tumble and swim).
3) Bacteria in regions with noxious substances or low-nutrient regions will die and disperse, respectively.

4) Bacteria in high-nutrient regions will reproduce by splitting. They will also attempt to attract other bacteria by generating chemical attractors.
5) Bacteria now disperse to seek new nutrient regions in the map.

From the above list, four processes can summarize the bacterial foraging behavior: (1) chemotaxis, (2) swarming, (3) reproduction and (4) elimination-dispersal. BFOA is precisely based on them [6].
A bacterium $\theta^i(j,k,l)$ represents a potential solution to the optimization problem, identified as $\vec{x}$ in Section I. $\theta^i(j,k,l)$, means that bacterium i is in its jth chemotactic loop, its kth reproduction loop, and its lth elimination-dispersal loop.

Within the chemotactic loop, a bacterium tumble was modeled by Passino [6] with a random search direction $\phi(i)$ as presented in Equation 1:

$$\phi(i) = \frac{\Delta(i)}{\sqrt{\Delta(i)^T\Delta(i)}} \quad (1)$$

where $\Delta(i)$ is a randomly generated vector of size n with elements within the following interval: $[-1,1]$. Afterwards, each bacterium i changes its position by a swim movement as indicated in Equation 2:

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\phi(i) \quad (2)$$

where $\theta^i(j+1,k,l)$ is the new position of bacterium i, $\theta^i(j,k,l)$ is the current position of bacterium i, $C(i)$ is the stepsize value defined by the user, and $\phi(i)$ is the tumble value as detailed previously in Equation 1.

The swim will be repeated N_s times if and only if the new position is better than the previous one, i.e., (assuming minimization) $f(\theta^i(j+1,k,l)) < f(\theta^i(j,k,l))$. Otherwise, a new tumble is computed. The chemotactic loop stops when the chemotactic loop limit N_c is reached for all bacteria in the swarm. The swarming process is based on a modification of the fitness landscape by making the boundaries of the location of a given bacterium more attractive, with the aim to attract more bacteria to such a region. This process requires the definition of a set of parameter values by the user. [6].

The reproduction process consists of sorting all bacteria in the swarm $\theta^i(j,k,l), \forall i,\ i=1,\ldots,N_b$ based on their objective function value $f(\theta^i(j,k,l))$ and eliminating half of them with the worst value. The remaining half will be duplicated so as to maintain a fixed swarm size.

The elimination-dispersal process consists on eliminating each bacteria $\theta^i(j,k,l), \forall i,\ i=1,\ldots,N_b$ with a probability $0 \leq P_{ed} \leq 1$.

The corresponding pseudocode is presented in Figure 1 where, in its caption, those BFOA input parameters are summarized.

III. Constraint-handling techniques

Recalling from Section I, BFOA, as other NIAs, was originally designed to sample unconstrained search spaces. A considerable amount of research has been devoted to design competitive constraint-handling techniques for NIAs [7], [16].

```
Begin
  Create a random initial swarm of bacteria $\theta^i(j,k,l)$
  $\forall i,\ i=1,\ldots,S_b$
  Evaluate $f(\theta^i(j,k,l))\ \forall i,\ i=1,\ldots,S_b$
  For l=1 to $N_{ed}$ Do
    For k=1 to $N_{re}$ Do
      For j=1 to $N_c$ Do
        For i=1 to $S_b$ Do
          Update $f(\theta^i(j,k,l))$ to emulate the swarming process
          Perform the chemotaxis process
          (tumble-swim) with Equations 1 and 2
          for bacteria $\theta^i(j,k,l)$ controlled by $N_s$
        End For
      End For
      Perform the reproduction process by sorting
      all bacteria in the swarm based on $f(\theta^i(j+1,k,l))$,
      deleting the $S_r$ worst bacteria and duplicating the remaining $S_b - S_r$
    End For
    Perform the elimination-dispersal process by eliminating
    each bacteria $\theta^i(j,k,l)\ \forall i,\ i=1,\ldots,S_b$ with probability $0 \leq P_{ed} \leq 1$
  End For
End
```

Fig. 1. BFOA pseudocode. Input parameters are number of bacteria S_b, chemotactic loop limit N_c, swim loop limit N_s, reproduction loop limit N_{re}, number of bacteria for reproduction S_r (usually $S_r = S_b/2$), elimination-dispersal loop limit N_{ed}, stepsizes C_i (they depend of the dimensionality of the problem) and probability of elimination dispersal P_{ed}.

Based on an updated review of the state-of-the-art in nature-inspired constrained optimization [9] the constraint-handling techniques can be divided in two generations. In the first one those constraint-handling mechanisms were focused on:

- Penalty functions [17]: based on decreasing the fitness of infeasible solutions depending on their constraint violation to favor feasible ones in the selection process. Penalty factors defined by the user set the severity of the punishment.
- Decoders [18]: based on a transformation of the original feasible region into a more suitable space where the search was carried out.
- Special operators[19]: designed for particular problems to maintain the feasibility of new solutions generated from feasible ones.
- Separation of objective function and constraints [20]: unlike penalty functions, where those values are combined into a single value, these approaches use such values separately as selection criteria depending of the situation in the search.

In those first-generation constraint-handling techniques different shortcomings were observed, such as an unsuitable bias (e.g. separation of objective functions and constraints), need of a careful fine-tuning of parameters (e.g. penalty functions), difficulties for generalization (e.g. special operators), or even high computational cost and/or difficult implementations (e.g. decoders).

The second-generation of constraint-handling techniques are the following:

- Feasibility rules [21]: a set of three criteria where the objective function and the sum of constraint violation are handled separately without adding additional user-defined parameters.

- Stochastic ranking [22]: a sorting mechanism based on feasibility; however, based on a user-defined parameter, some solutions are sorted based only in their objective function value, regardless of feasibility.
- ϵ-Constrained method [23]: transforms a CNOP into an unconstrained numerical optimization problem by using a tolerance called ϵ to consider as feasible those slightly infeasible solutions.
- Novel penalty functions [24], [25]: mostly dynamic or adaptive penalty functions which may not require user-defined penalty factors.
- Novel special operators [26]: unlike those of the previous generation, these ones are focused on sampling regions of interest in the search space, e.g., the boundaries of the feasible region.
- Multi-objective concepts [27]: transforms the CNOP into an unconstrained multi-objective optimization problem where Pareto dominance or Pareto ranking are used as selection criteria.
- Ensemble of constraint-handling techniques [28]: a combination of two or more constraint-handling techniques within a NIA.

Another issue taken into account in those second-generation constraint-handling techniques is the importance of the search algorithm adopted. One of the first NIAs used were the genetic algorithm [19], [29], [30], DE [31], [32], [33] and PSO [34], [35], [36] among others. Motivated precisely by such preference for some particular NIAs, this paper surveys BFOA adapted to solve CNOPs, as will be detailed in the next section of this document.

IV. Implementation of BFOA in problems with constraints

In this section those BFOA versions which solve CNOPs are presented and discussed.

Mezura-Montes and Hernández-Ocaña in [37] proposed the modified bacterial foraging optimization algorithm (MBFOA) to solve CNOPs, particularly mechanical design problems. The modifications can be divided in two aspects: (1) the simplification of the original BFOA (i.e. decreasing the number of user-defined parameters and eliminating one of the four nested *for* loops, see Figure 1) and (2) adding a suitable constraint-handling mechanism. As a result, in MBFOA each bacterium performed its own chemotactic loop within a generation loop and the reproduction and elimination-dispersal loops were changed as single steps. Moreover, the stepsize value was defined by using a formula based on the limits of the design variables instead of asking the user to tune it. A simple swarming mechanism was also added to the redefined chemotactic step for each bacterium in the population, i.e., at half and at the end of the chemotactic loop, instead of a tumble-swim movement, each bacterium is attracted by the best bacterium in the swarm. Regarding constraint-handling, the feasibility rules [21] (second-generation constraint-handling technique) were adopted as criteria in the comparison of solutions so as to bias the search to the feasible region. The approach was tested on three well-known engineering design problems. The results were compared against PSO variants. MBFOA provided competitive results with a lower number of function evaluations while requiring less parameters to be defined by the users. The main advantages of MBFOA is the reduction of user-defined parameters and one inner loop of BFOA. However, the approach was particularly sensitive to the stepsize value, even it was defined by using the boundaries of the decision variables. Finally, the feasibility rules allowed MBFOA to consistently reach the feasible region of the search space, but their strong preference for feasible solutions led to premature convergence in some runs.

MBFOA was later extended by Mezura-Montes et al. [38] to solve a bi-objective mechanical design optimization problem of a continuously variable transmission, where the output of a four-bar mechanism must be maximized while the differences of the transmission angles must be minimized. Besides inequality and equality constraints, the bi-objective problem considered in its model one dynamic inequality constraint related with an angle which must maintain its value below a specific value along a whole crank cycle. Pareto dominance coupled with the set of feasibility rules (second-generation constraint-handling technique) were adopted as selection criteria in the approach. Crowding distance was added as a diversity mechanism in the objective space, while elitism by an external archive was considered and the reproduction of bacteria was used at a minimal basis. The results obtained were compared with respect to those of four state-of-the-art approaches for multi-objective optimization. The main advantage of the approach was that the Pareto front obtained was clearly different from those reached by the compared algorithms i.e. MBFOA was able to sample different (and more suitable for the problem based on later experiments) regions of the search space. Furthermore, the feasible region of the search space was consistently reached even in presence of the dynamic constraint. In fact, feasible solutions were found early in the search. However, the algorithm was sensitive to its parameters, particularly the stepsize value. Moreover, the performance based on metrics such as two-set coverage and hypervolume was not so competitive.

Praveena et al.[11] proposed a hybrid approach BFOA-PSO-DE to solve a dynamic economic dispatch problem. The velocity value used in PSO plus the differential mutation of DE were applied to bacteria within and after the traditional tumble-swim operator, respectively. Instead of using a random search direction, bacteria were guided by a velocity vector as in PSO. After such movement, the differential mutation operator was applied to each bacterium. Regarding the BFOA parameters, the stepsize was randomly chosen between a range $(0.1, 1.8)$. A first-generation constraint-handling technique based on a static penalty function (i.e., a penalty factor is defined at the beginning of the search and remains fixed during the process) was adopted. From the results reported it is assumed that such a constraint-handling technique provided a competitive performance because only feasible solutions were shown and no significant sensitivity to the lone penalty factor (a value of 0.1 or 0.2) was reported. A comparison against other hybrid approaches suggested a competitive performance of the proposed algorithm. However, no statistical information of the results was provided nor further details on the number of independent runs carried out. Finally an important shortcoming of the approach is the high number of parameters, most of them inherited by PSO and DE, that must be fine-tuned by the user, including the penalty factor which worked well, but only one problem was solved.

Chunguang et al. [39] implemented BFOA to solve a smelting ingredient diluting optimization problem. Unlike other BFOA applications, this approach kept intact all parts of the algorithm. A penalty function based on Lagrange multipliers (first-generation constraint-handling technique) was adopted to incorporate feasibility information in the selection processes within BFOA. The results obtained with BFOA in five independent runs were better compared with those obtained by a heuristic experience method. The main advantages of the approach are the obtained results and the simplicity of the constraint-handling technique adopted. However, the BFOA parameter values, including the stepsizes, were defined by the user and only one test instance was tackled. Therefore, the sensitivity of BFOA to its parameters, including those related with the constraint-handling technique, is unknown. Finally, no comparison against other NIAs were reported.

Wei et al. [40] implemented a BFOA variant to solve one instance of the thermal non-destructive test and evaluation (TNDT/E) problem to identify parameters with defects. A surrogate model based on a radial basis function neural network was used to decrease the computational time required to evaluate a single solution. A first-generation constraint-handling technique based on death-penalty (i.e., an infeasible solution is eliminated and another one is generated at random until it is feasible) was adopted in the approach. A comparison was made against a PSO-based algorithm. Even the approach seems to provide a suitable convergence behavior based on the plots shown in the paper with respect to that obtained by PSO, no statistical information nor validation was presented. Therefore, the robustness of the approach was not analyzed. On the other hand, considering that no further comments were made regarding the difficulty to generate feasible solutions, the feasible region of the search space was not hard to find even with the simple constraint-handling adopted.

Pangrahi et al. [10] adapted BFOA to solve a bi-objective instance of the economic emission dispatch problem. Two sorting mechanisms taken from evolutionary multi-objective optimization were added to BFOA to deal with two-objectives instead of only one (non-dominated sorting and fuzzy-dominance-based sorting). Those sorting mechanisms were used to select the bacteria to remain for the next cycle. Moreover, a parameter called "guide" was added so as to allow some bacteria to swim by using a direction previously used by other bacteria. The equality constraints of the problem were satisfied by a special mechanism using the Newton-Raphson method. No details were given on the constraint-handling for the inequality constraints of the problem. However, the results provided were feasible. A fuzzy mechanism for an *a-posteriori* preference handling was used to select the most suitable solution from the final Pareto front obtained. A comparison of results against well-known evolutionary multi-objective algorithms such as NSGA, NPGA, SPEA, among others, was presented, where those from the BFOA version with fuzzy-dominance-based sorting were the most competitive. Furthermore, the results on three performance metrics showed that the fuzzy-dominance-based sorting was superior with respect to non-dominated sorting in BFOA. The main disadvantages of the approach are the very specific type of constraint-handling mechanism which is difficult to generalize and also the high number of parameter values which must be defined by the user. In fact, the authors remark that the stepsize required by the tumble operator is one of the most sensitive parameters in the algorithm. Finally, only one test instance was solved.

Pangrahi et al. [41] revisited the non-dominated sorting mechanism in BFOA while solving the same problem of their previous work (the economic emission dispatch problem) with an improvement related with the "guide" parameter. In this more recent work, the search direction used from previous bacteria was that of a bacterium with a better rank with respect to the current bacteria (the one performing its tumble-swim movement). Constraints were handled in a similar way to the previous work. Finally, a comparison against a classic multi-objective optimization approach called ϵ-constrained method, was included. The advantages of this new version of the approach remain the same i.e., the competitive results. In the same way, the shortcomings of the new version are inherited from the former (the very specific constraint-handling mechanism, the high number of parameter values, including the "guide" value, to be defined by the user and the fact that only one problem was solved).

Deshpande et al. [42] adopted BFOA to solve a supply chain optimization problem modelled as an improved fuzzy multi-objective optimization problem which was transformed into a single-objective optimization problem by using a weighted sum approach. The approach minimized the operation cost and the delivery time of the product, while maximizing the level of index service to take a control of the product inventory in the supply chain. Constraints were handled with a first-generation constraint-handling technique based on death-penalty, where infeasible solutions were eliminated and new solutions were generated at random until they were feasible, i.e. the search was carried out only within the feasible region of the search space. The main change in BFOA was the solution encoding, because the decision variables took integer values instead of real values (as in the original BFOA). Therefore, a rounding process was made to the continuous values of each bacterium before applying the tumble-swim operator. Two cases of one test instance were solved by the approach and the results obtained were competitive in both of them. Such performance seems to be the main advantage of the approach. However, it was not clear how difficult it was to generate feasible solutions at random. Furthermore, the multi-objective problem was solved as a single-objective one, which may cause some trade-off solutions not being generated due to the weight values adopted in the experiments. Finally, no extensive statistical validation was provided.

Mezura-Montes and López-Dávila [43] improved the modified BFOA (MBFOA) with an adaptive parameter control mechanism for the stepsize, a reduction in the frequency of the reproduction and elimination-dispersal processes and the addition of a local search operator when solving CNOPs. The stepsize, which has been reported as a critical parameter for BFOA, was adapted based on the successful movements made by all bacteria in a cycle. If the overall success movement percentage was below 20%, the stepsize values were decreased because convergence was being reached. Otherwise, the stepsize values were increased to promote more exploration. As in MBFOA, the initial stepsize values were defined according to the design variables boundaries. A mathematical programming method called Hooke-Jeeves was adopted as a local search operator. It was applied to a user-defined percentage of the best bacteria in the swarm (based on the second-generation constraint-handling technique known as the feasibility rules) at certain cycles (defined by the user). Also at certain cycles the reproduction and elimination-dispersal processes were applied. The proposal (known as MBFOA-Adaptive Stepsize and Local Search, MBFOA-AS-LA for short) was tested in a well-known benchmark for constrained numerical optimization and compared with respect to BFOA and other NIAs. Despite the fact that the usage of local search decreased the evaluations required by the algorithm to reach competitive results, premature convergence was observed in some test problems and also a high sensitivity to some parameters (one related with the initial stepsize and another related with the swarming operator) was reported.

Pandit et al. [12] proposed a μ-BFOA to solve four instances of the environmental economic dispatch problem. In their approach, a swarm of only three bacteria was used and the stepsize was adapted with an oscillatory behavior during the optimization run. From the three bacteria in the swarm, the best one was kept for the next iteration, the second best was reproduced with a crossover operator, while the third one was eliminated and replaced with a new bacterium generated at random. A first-generation static penalty function was used as a constraint-handling mechanism. The approach was compared against the original BFOA, the well-known genetic algorithm for multi-objective optimization called NSGA-II, an artificial immune system, and PSO, among others. The proposal showed a competitive performance even working with a swarm of only three bacteria and this is precisely its main advantage, i.e. a low computational cost may be obtained. However, the μ-BFOA inherited the drawback of a static penalty function because its fine-tuning must be made if the approach is applied to other problems. Furthermore, a new parameter was added for the crossover operator adopted for the second best bacterium in the swarm.

Vaisakh et al. [44] added operators from PSO and DE into BFOA to solve static and dynamic instances of the economic dispatch optimization problem. The tumble operator was replaced by a velocity calculation as in PSO. Furthermore, after the chemotactic step, the differential mutation was used to modify the position of bacteria. The stepsize was generated at random between suitable values $[0.1, 5.0]$ during a single run. A first-generation static penalty function was used to deal with the constraints of the problem. The results obtained were compared against the original BFOA, ED, PSO, and memetic algorithms with sequential quadratic programming (SQP) as local search operator, e.g. evolutionary programming with SQP and PSO with SQP. The overall performance of the approach was very competitive. The main advantage of the approach is the enriched velocity-swim operator which improved its convergence behavior. Furthermore, the differential mutation helped the algorithm to avoid local optimum solutions. However, besides the fine-tuning required by the approach for the penalty factors adopted (four in this case) additional parameter values must be defined by the user, e.g. from PSO the acceleration coefficients c_1, c_2, and the inertia factor w, and from ED the amplification factor s_f used in the differential mutation.

Saber [45] included BFOA's tumble-swim operator into PSO to avoid premature convergence when solving economic dispatch optimization problems. A first-generation static penalty function was used as a constraint-handling mechanism. However, an interesting usage of the amount of constraint violation was proposed in this work, because it was used, coupled with the iteration number, to adaptively define the stepsize of the tumble-swim operator, i.e. the feasibility information biased the movement of bacteria. The approach was compared against an evolutionary algorithm with and without SQP as a local search operator. The main advantage of the approach is its ability to provide better results with less computational cost. It is also remarkable the way the feasibility information is used to control the stepsize of bacteria in the search. The main shortcomings of the approach are the limited number of test problems solved (i.e., questions about its performance in other instances remain without an answer), and the number of parameters (PSO and BFOA) which must be fine-tuned by the user, including the penalty factors.

Niu et al. [46] proposed a linear decreasing stepsize value within BFOA to solve portfolio optimization problems. A first-generation static penalty function was used for constraint-handling. Preliminary experiments in some well-known unconstrained numerical optimization problems showed the ability of the linear-decreasing mechanism to avoid premature convergence. Furthermore, based on the statistical results provided by the approach in the portfolio optimization problems solved, its performance was clearly better with respect to the traditional BFOA, a PSO with also a linear decreasing mechanism for the inertia weight value, and also with respect to a traditional genetic algorithm. Such performance, besides the simplicity of the linear decreasing mechanism, are the main advantages of the approach. However, the main shortcoming, besides the static penalty function adopted, is the definition of two additional parameters precisely related with the linear decreasing mechanism for the stepsize value, whose values might be problem-dependent.

V. Analysis

From the review presented in Section IV and the summary in Table I, the following findings are outlined:

A. *Constraint-handling*

Most algorithms based on BFOA to solve CNOPs use first-generation constraint-handling techniques. The most popular has been the static penalty function [11], [12], [39], [44], [45], [46], followed by the death penalty for some problems where it was not so difficult to find the feasible region of the search space [40], [42], and special operators particularly related with the economic dispatch optimization problem [10], [41]. Regarding second-generation constraint-handling techniques, the set of feasibility rules has been the only one adopted by BFOA [37], [38], [43].

B. *Modifications to BFOA*

- The most critical parameter of BFOA when solving CNOPs is the stepsize. However, in most of the approaches discussed in Section IV, it was a user-defined parameter [10], [39], [40], [41], [42]. On the other hand, there are initial efforts to design expressions to assign stepsize values based on the features of the problem [37], [38]. There are also less popular proposals to (1) generate stepsize values at random [11], [44], (2) use dynamic adjustment [12], [46], or (3) use adaptive approaches [43], [45]. It is worth remarking that Saber [45] proposed an adaptive mechanism for the stepsize which uses the feasibility information in such calculations.
- Another improvement made to BFOA when solving CNOPs was on the tumble-swim operator, e.g., a velocity instead of a random search direction [11], [44], a guide mechanism [10], [41], and a swarming operator [37], [38], [43].
- The reproduction process was minimized-modified in BFOA to avoid premature convergence [10], [12], [38], [40], [41], [43], [45].
- BFOA has been improved by combining it with other algorithms such as PSO-DE [11], [44], only PSO [45], and Hooke-Jeeves as a local search operator [43].
- There are BFOA variants with small populations (i.e. μ-BFOA) [12] as well as an adapted version for discrete search spaces [42].

C. *Problems solved*

It was interesting to note that the most tackled problem was the economic dispatch optimization problem [10], [11], [12], [41], [44], [45], followed by mechanical design instances [37], [38], engineering processes [39], [40], supply chain optimization [42], and portfolio optimization [46]. Quite surprising, only one approach has been tested in well-known benchmarks for CNOPs [43].

D. *Promising paths of research*

From the above discussion, the following paths of research are considered as promising for BFOA to solve CNOPs:

- The addition of second-generation constraint-handling mechanisms (e.g. ϵ-constrained, multi-objective concepts, stochastic ranking, ensemble of constraint-handling techniques, among others) to BFOA.
- The definition of other dynamic or adaptive mechanisms for the stepsize value, where the feasibility information is considered, so as to make them more suitable for constrained optimization.
- Solving a wider set of test problems [47], [48], where performance measures are computed [49].
- Combining BFOA with other NIAs such as evolution strategies, artificial bee colony [49], evolutionary programming [50], genetic algorithms [1], etc.
- Considering local search operators, already successfully used by other NIAs, now added to BFOA (e.g. SQP, Nealder-Mead).
- Adopting other crossover operators for the reproduction process.
- Further studies of surrogates models in BFOA when dealing with constrained search spaces.
- A more in-depth analysis of both, the BFOA for discrete search spaces and the μ-BFOA.

VI. Conclusions

This paper presented a review of the bacterial foraging optimization algorithm used to solve constrained numerical optimization problems. After a brief introduction to the algorithm and its elements, a classification of constraint-handling techniques taken from the specialized literature was presented and adopted to describe and discuss each BFOA approach. From the literature review, a lack of novel constraint-handling techniques in BFOA was observed, i.e., mostly first-generation penalty functions and special operators have been adopted, then more second-generation techniques may be included in future works. Moreover, different elements of BFOA have been subject of improvement and more work on them is suggested (stepsize value, search direction in the tumble-swim operator and reproduction process). Even BFOA has been hybridized, mainly with PSO and DE, other NIAs may be used with that purpose. The same applies for local search operators and also surrogate models. Those novel versions of BFOA for discrete search spaces and also the μ-BFOA deserve more study and analysis. Finally, BFOA is still a young algorithm compared with, for example, EAs, and must be analyzed in more test problems using performance measures. As a final conclusion, BFOA seems to be in the very initial phase in its usage to solve CNOPs. Therefore, more research work is expected in the years to come so as to get another solid option to deal with numerical optimization problems in presence of constraints.

TABLE I
MAIN FEATURES OF EACH BFOA APPROACH DIVIDED BY THE GENERATION OF THE CONSTRAINT-HANDLING ADOPTED.

Authors	**Constraint-handling technique**	**Step size handling**	**Special feature**
First generation			
Praveena et al. [11]	Penalty function	Random	Hybrid BFOA-PSO-DE
Chunguang et al. [39]	Penalty function	User-defined	None
Wei et al. [40]	Death penalty	User-defined	Surrogate-assisted
Pangrahi et al. [10]	Special operator	User-defined	Tumble-swim operator improvement
Pangrahi et al. [41]	Special operator	User-defined	Tumble-swim operator improvement
Deshpande et al. [42]	Death penalty	User-defined	Integer search space
Pandit et al. [12]	Penalty function	Dynamic	Small population
Vaisakh et al. [44]	Penalty function	Random	Hybrid BFOA-PSO-DE
Saber [45]	Penalty function	Adaptive using feasibility information	Hybrid PSO-BFOA
Niu et al. [46]	Penalty function	Dynamic	None
Second generation			
Mezura-Montes and Hernández-Ocaña [37]	Feasibility rules	Defined by formula and fixed during the run	Simplification of the original BFOA
Mezura-Montes et al. [38]	Feasibility rules	Defined by formula and fixed during the run	Simplification of the original BFOA for multi-objective optimization
Mezura-Montes and López-Dávila [43]	Feasibility rules	Adaptive	Local search

ACKNOWLEDGMENT

The first author acknowledges support from CONACyT and UJAT through a scholarship to pursue PhD studies at the Universidad Juárez Autónoma de Tabasco.

REFERENCES

[1] A. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, ser. Natural Computing. Springer Verlag, 2003.

[2] A. P. Engelbrecht, *Fundamentals of Computational Swarm Intelligence*. John Wiley & Sons, 2005.

[3] P. Siarry and Z. Michalewicz, Eds., *Advances in Metaheuristic Methods for Hard Optimization*. Berlin: Springer, 2008, iSBN 978-3-540-72959-4.

[4] J. Kennedy, R. C. Eberhart, and Y. Shi, *Swarm Intelligence*. UK: Morgan Kaufmann, 2001.

[5] D. Karaboga and B. Basturk, "powerful and efficient algorithm for numerical function optimization: Artificial bee colony (abc) algorithm," *Journal of Global Optimization*, vol. 39, no. 3, pp. 459–471, 2007.

[6] K. Passino, "Biomimicry of bacterial foraging for distributed optimization and control," *IEEE Control Systems Magazine*, vol. 22, no. 3, pp. 52–67, 2002.

[7] E. Mezura-Montes, Ed., *Constraint-Handling in Evolutionary Optimization*, ser. Studies in Computational Intelligence. Springer-Verlag, 2009, vol. 198.

[8] E. Mezura-Montes and J. I. Flores-Mendoza, "Improved particle swarm optimization in constrained numerical search spaces," in *Nature-Inspired Algorithms for Optimization*, R. Chiong, Ed. Springer-Verlag, Studies in Computational Intelligence Series, 2009, ISBN: 978-3-540-72963-1., 2009, vol. 193, pp. 299–332.

[9] E. Mezura-Montes and C. A. C. Coello, "Constraint-handling in nature-inspired numerical optimization: Past, present and future," *Swarm and Evolutionary Computation*, vol. 1, no. 4, pp. 173–194, 2011.

[10] B. Panigrahi, V. R. Pandi, S. Das, and S. Das, "Multiobjective fuzzy dominance based bacterial foraging algorithm to solve economic emmission dispatch dispatch problem," *Energy*, vol. 35, no. 12, pp. 4761–4770, 2010.

[11] P. Praveena, K. Vaisakh, and S. R. M. Rao, "A bacterial foraging and pso-de algorithm for solving dynamic economic dispatch problem with valve-point effects," in *First International Conference on Integrated Intelligent Computing*. IEEE Press, 2010, pp. 227–232.

[12] N. Pandit, A. Tripathi, S. Tapaswi, and M. Pandit, "An improved bacterial foraging algorithm for combined static/dynamic enviromental economic dispatch," *Applied Soft Computing*, vol. 0, no. 12, pp. 3500–3513, 2012.

[13] H. Chen, Y. Zhu, and K. Hu, "Cooperative bacterial foraging optimization," *Discrete Dynamics in Nature and Society*, vol. 2009, pp. 1–17, 2009.

[14] P. D. Sathya and R. Kayalvizhi, "Optimal multilevel thresolding using bacterial foraging algorithm," *Expert Systems with Applications*, vol. 0, no. 38, pp. 15 549–15 564, 2011.

[15] S. Kamyab and A. Bahrololoum, "Designing of rule base for a tsk-fuzzy system using bacterial foraging optimization algorithm (bfoa)," *Social and Behavioral Sciences*, vol. 0, no. 32, pp. 176–183, 2012.

[16] Z. Michalewicz and M. Schoenauer, "Evolutionary Algorithms for Constrained Parameter Optimization Problems," *Evolutionary Computation*, vol. 4, no. 1, pp. 1–32, 1996.

[17] A. E. Smith and D. W. Coit, "Constraint Handling Techniques—Penalty Functions," in *Handbook of Evolutionary Computation*, T. Bäck, D. B. Fogel, and Z. Michalewicz, Eds. Oxford University Press and Institute of Physics Publishing, 1997, pp. C5.2–1–C5.2–6.

[18] S. Koziel and Z. Michalewicz, "Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization," *Evolutionary Computation*, vol. 7, no. 1, pp. 19–44, 1999.

[19] Z. Michalewicz and G. Nazhiyath, "Genocop III: A co-evolutionary algorithm for numerical optimization with nonlinear constraints," in *Proceedings of the Second IEEE International Conference on Evolutionary Computation*, D. B. Fogel, Ed. Piscataway, New Jersey: IEEE Press, 1995, pp. 647–651.

[20] M. Schoenauer and S. Xanthakis, "Constrained GA Optimization," in

Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA-93), S. Forrest, Ed., University of Illinois at Urbana-Champaign. San Mateo, California: Morgan Kauffman Publishers, July 1993, pp. 573–580.

[21] K. Deb, "An Efficient Constraint Handling Method for Genetic Algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2–4, pp. 311–338, 2000.

[22] T. P. Runarsson and X. Yao, "Stochastic Ranking for Constrained Evolutionary Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 3, pp. 284–294, September 2000.

[23] T. Takahama and S. Sakai, "Constrained Optimization by the ϵ Constrained Differential Evolution with Gradient-Based Mutation and Feasible Elites," in *2006 IEEE Congress on Evolutionary Computation (CEC'2006)*. Vancouver, BC, Canada: IEEE, July 2006, pp. 308–315.

[24] B. Tessema and G. G. Yen, "An adaptive penalty formulation for constrained evolutionary optimization," *IEEE Transactions on Systems, Man and Cybernetics Part A—Systems and Humans*, vol. 39, no. 3, 2009.

[25] N. Jin, E. Tsang, and J. Li, "A constraint-guided method with evolutionary algorithms for economic problems," *Applied Soft Computing*, vol. 9, no. 3, pp. 924–935, 2009.

[26] G. Leguizamón and C. A. C. Coello, "Boundary Search for Constrained Numerical Optimization Problems with an Algorithm Inspired on the Ant Colony Metaphor," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 350–368, 2009.

[27] E. Mezura-Montes and C. A. Coello-Coello, "Constrained optimization via multiobjective evolutionary algorithms," in *Multiobjective Problems Solving from Nature: From Concepts to Applications*, D. C. in Joshua Knowles and K. Deb, Eds. Springer-Verlag, Natural Computing Series, 2008, ISBN: 978-3-540-72963-1., 2008, pp. 53–76.

[28] R. Mallipeddi and P. N. Suganthan, "Ensemble of Constraint Handling Techniques," *IEEE Transactions On Evolutionary Computation*, vol. 14, no. 4, pp. 561–579, August 2010.

[29] M. Schoenauer and Z. Michalewicz, "Evolutionary Computation at the Edge of Feasibility," in *Proceedings of the Fourth Conference on Parallel Problem Solving from Nature (PPSN IV)*, H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, Eds., Berlin, Germany. Heidelberg, Germany: Springer-Verlag, September 1996, pp. 245–254.

[30] R. Kowalczyk, "Constraint Consistent Genetic Algorithms," in *Proceedings of the 1997 IEEE Conference on Evolutionary Computation*. Indianapolis, USA: IEEE, April 1997, pp. 343–348.

[31] J. J. Liang, S. Zhigang, and L. Zhihui, "Coevolutionary comprehensive learning particle swarm optimizer," in *2010 Congress on Evolutionary Computation (CEC'2010)*. Barcelona, Spain: IEEE Service Center, July 2010, pp. 1505–1512.

[32] E. Mezura-Montes, C. A. Coello Coello, and E. I. Tun-Morales, "Simple Feasibility Rules and Differential Evolution for Constrained Optimization," in *Proceedings of the 3rd Mexican International Conference on Artificial Intelligence (MICAI'2004)*, R. Monroy, G. Arroyo-Figueroa, L. E. Sucar, and H. Sossa, Eds. Heidelberg, Germany: Springer Verlag, April 2004, pp. 707–716, lecture Notes in Artificial Intelligence No. 2972.

[33] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. C. Coello, "Promising Infeasibility and Multiple Offspring Incorporated to Differential Evolution for Constrained Optimization," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2005)*, H.-G. Beyer, U.-M. O'Reilly, D. Arnold, W. Banzhaf, C. Blum, E. Bonabeau, E. Cant Paz, D. Dasgupta, K. Deb, J. Foste r, E. de Jong, H. Lipson, X. Llora, S. Mancoridis, M. Pelikan, G. Raidl, T. Soule, A. Tyrrell, J.-P. Watson, and E. Zitzler, Eds., vol. 1, Washington DC, USA. New York: ACM Press, June 2005, pp. 225–232, iSBN 1-59593-010-8.

[34] K. Zielinski and R. Laur, "Constrained Single-Objective Optimization Using Particle Swarm Optimization," in *2006 IEEE Congress on Evolutionary Computation (CEC'2006)*. Vancouver, BC, Canada: IEEE, July 2006, pp. 1550–1557.

[35] C.-L. Sun, J.-C. Zeng, and J.-S. Pan, "A Particle Swarm Optimization with Feasibility-Based Rules for Mixed-Variable Optimization Problems," in *Ninth International Conference on Hybrid Intelligent Systems, 2009. HIS '09*, J.-S. Pan, J. Liu, and A. Abraham, Eds. Shenyang, China: IEEE Computer Society Press, August 2009, pp. 543–547.

[36] Q. He and L. Wang, "A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization," *Applied Mathematics and Computation*, vol. 186, no. 2, pp. 1407–1422, March 15th 2007.

[37] E. Mezura-Montes and B. Hernández-Ocaña, "Modified bacterial foraging optimization for engineering design," in *Proceedings of the Artificial Neural Networks in Enginnering Conference (ANNIE'2009)*, ser. Intelligent Engineering Systems Through Artificial Neural Networks, C. H. Dagli and et al., Eds., vol. 19. St. Louis, MO, USA: ASME Press, November 2009, pp. 357–364.

[38] E. Mezura-Montes, E. A. Portilla-Flores, and B. Hernández-Ocaña, "Optimum synthesis of a four-bar mechanism using the modified bacterial foraging algorithm," *International Journal of Systems Science*, 2013, DOI:10.1080/00207721.2012.745023.

[39] C. Chunguang, Z. Yunlong, H. Kunyuan, and S. Hai, "Research on smelting ingredient diluting for refined copper by bacterial foraging optimization algorithm," in *International Conference on Digital Manufacturing and Automation*. IEEE, 2010, pp. 275–278.

[40] K. Wei, S. Feng-rui, Y. Li, and C. Lin-gen, "Application of improved bcc algorithm and rbfnn in identification of defect parameters," in *Fourth International Conference on Genetic and Evolutionary Computing*. IEEE, 2010, pp. 160–164.

[41] B. Panigrahi, V. R. Pandi, R. Sharma, S. Das, and S. Das, "Multiobjetive bacteria foraging algorithm for electrical load dispatch problem," *Energy Conversion and Management*, vol. 52, no. 2, pp. 1334–1342, 2011.

[42] P. Deshpande, D. Shukla, and M. Tiwari, "Fuzzy goal programming for inventory management: A bacterial foraging approach," *European Journal of Operational Research*, vol. 0, no. 212, pp. 325–336, 2011.

[43] E. Mezura-Montes and E. A. López-Davila, "Adaptation and local search in the modified bacterial foraging algorithm for constrained optimization," in *Proccedings of the IEEE Congress on Evolutionary Computation 2012*. ISBN:978-1-4673-1508-1, 2012, pp. 497–504.

[44] K. Vaisakh, P. Praveena, S. R. M. Rao, and K. Meah, "Solving dynamic economic dispatch problem with security constraints using bacterial foraging pso-de algorithm," *Electrical Power and Energy Systems*, vol. 0, no. 39, pp. 56–67, 2012.

[45] A. Y. Saber, "Economic dispatch using particle swarm optimization with bacterial foraging effect," *Electrical Power and Energy Systems*, vol. 0, no. 34, pp. 38–46, 2012.

[46] B. Niu, Y. Fan, H. Xiao, and B. Xue, "Bacterial foraging based approaches to portfolio optimization with liquidity risk," *Neurocomputing*, vol. 98, no. 0, pp. 90 – 100, 2012. [Online]. Available: http://www.sciencedirect.com/science/ar

[47] J. J. Liang and P. N. Suganthan, "Dynamic Multi-Swarm Particle Swarm Optimizer with a Novel Constrain-Handling Mechanism," in *2006 IEEE Congress on Evolutionary Computation (CEC'2006)*. Vancouver, BC, Canada: IEEE, July 2006, pp. 316–323.

[48] R. Mallipeddi and P. N. Suganthan, "Problem definitions and evaluation criteria for the cec 2010 competition on constrained real-parameter optimization," Nanyang Technological University, Singapore, Tech. Rep., 2010.

[49] E. Mezura-Montes and O. Cetina-Domíngez, "Empirical analysis of a modified artificial bee colony for constrained numerical optimization," *Applied Mathematics and Computation*, vol. 218, no. 22, pp. 10 943–10 973, 2012.

[50] L. J. Fogel, *Intelligence Through Simulated Evolution. Forty years of Evolutionary Programming*. New York: John Wiley & Sons, 1999.