A Dynamic Hybrid Framework for Constrained Evolutionary Optimization

Yong Wang, *Member, IEEE*, and Zixing Cai, *Senior Member, IEEE*

***Abstract*—Based on our previous work, this paper presents a dynamic hybrid framework, called DyHF, for solving constrained optimization problems. This framework consists of two major steps: global search model and local search model. In the global and local search models, differential evolution serves as the search engine, and Pareto dominance used in multiobjective optimization is employed to compare the individuals in the population. Unlike other existing methods, the above two steps are executed dynamically according to the feasibility proportion of the current population in this paper, with the purpose of reasonably distributing the computational resource for the global and local search during the evolution. The performance of DyHF is tested on 22 benchmark test functions. The experimental results clearly show that the overall performance of DyHF is highly competitive with that of a number of state-of-the-art approaches from the literature.**

***Index Terms*—Constrained evolutionary optimization, constraint-handling technique, dynamic hybrid framework (DyHF), multiobjective optimization.**

I. Introduction

In GENERAL, the *constrained optimization problems* (COPs) can be formulated as follows: find the vector $\vec{x} = (x_1, \ldots, x_n) \in \Re^n$ which satisfies the l inequality constraints:

$$g_j(\vec{x}) \le 0, \quad j = 1, \ldots, l \tag{1}$$

and the $(m-l)$ equality constraints

$$h_j(\vec{x}) = 0, \quad j = l+1, \ldots, m \tag{2}$$

and minimizes the objective function $f(\vec{x})$, where $\vec{x} \in \Omega \subseteq S$, Ω is the feasible region defined by the m linear or nonlinear constraints

$$\Omega = \{\vec{x} \in S | g_j(\vec{x}) \le 0,\ j = 1, \ldots, l; \quad h_j(\vec{x}) = 0, j = l+1, \ldots, m\} \tag{3}$$

and S is the decision space defined by the parametric constraints

$$L_i \le x_i \le U_i, \quad 1 \le i \le n. \tag{4}$$

The inequality constraint that is equal to zero, i.e., $g_j(\vec{x}) = 0\ (j \in \{1, \ldots, l\})$, at any point $\vec{x} \in \Omega$ is called *active* constraint at $\vec{x}$. All the equality constraints $h_j(\vec{x})\ (j = l+1, \ldots, m)$ are considered *active* at all points of Ω.

In general, the degree of constraint violation of a vector $\vec{x}$ on the jth constraint is defined as

$$G_j(\vec{x}) = \begin{cases} \max\{0, g_j(\vec{x})\}, & 1 \le j \le l \\ \max\{0, |h_j(\vec{x})| - \delta\}, & l+1 \le j \le m \end{cases} \tag{5}$$

where δ is a positive tolerance value for equality constraints. Then, $G(\vec{x}) = \sum_{j=1}^{m} G_j(\vec{x})$ reflects the degree of constraint violation of the vector $\vec{x}$.

Evolutionary algorithms (EAs) are population-based search methods that take their inspiration from natural selection and survival of the fittest in the biological world. Facts have proved that EAs are very suitable to handle complicated COPs. The use of EAs for COPs has significantly grown in the past decade, giving rise to a large number of constrained optimization evolutionary algorithms (COEAs) [1]–[3].

COEAs consist of two major components: the search algorithm and the constraint-handling technique [3], [4]. Thus, the performance of COEAs is primarily dependent on these two components. The aim of the search algorithm is to adjust the exploration and exploitation abilities of the population, while the constraint-handling technique focuses on how to incorporate the constraints into the evolutionary process. The most popular constraint-handling techniques used in COEAs involve: methods based on penalty functions, methods based on biasing feasible over infeasible solutions, and methods based on multiobjective optimization concepts [1]–[3].

In [4], a hybrid EA called HCOEA is proposed by the authors to solve 13 benchmark test functions, which consists of both the global and local search models. However, the main drawback of HCOEA is that the performance of HCOEA is very sensitive to the problem-dependent expanding factor in the simplex crossover [5]. A trial-and-error process has to be used to choose a proper value for this expanding factor for the problems at hand, which limits the real-world applications of HCOEA. Moreover, HCOEA fails to solve some complex COPs such as the second, seventh, and tenth test functions used in [4].

To overcome the above drawbacks and further improve the performance of HCOEA, we generalize the idea of HCOEA

Manuscript received May 2, 2010; revised December 3, 2010, April 7, 2011, and June 23, 2011; accepted June 29, 2011. Date of publication August 4, 2011; date of current version December 7, 2011. This work was supported in part by the National Natural Science Foundation of China, under Grants 60805027 and 90820302, and in part by the Research Fund for the Doctoral Program of Higher Education of China, under Grant 200805330005. This paper was recommended by Associate Editor J. Wang.

The authors are with the School of Information Science and Engineering, Central South University, Changsha 410083, China (e-mail: ywang@csu.edu.cn; zxcai@csu.edu.cn).

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/TSMCB.2011.2161467

and propose a *dynamic hybrid framework* called DyHF in this paper, for constrained evolutionary optimization. Compared with HCOEA, DyHF has the following features: 1) the global and local search models are dynamically applied by tracking the feasibility proportion of the current population; 2) differential evolution (DE) serves as the search engine to generate the offspring population during both the global and local search process; and 3) the parameter settings are kept the same for different problems. DyHF has been conducted to solve 22 benchmark test functions. The experimental results indicate that DyHF is performed quite well since it can solve all the test functions.

The rest of this paper is organized as follows. Since DE is used as the search algorithm in this paper, it is briefly introduced in Section II. Section III briefly reviews the related work. Section IV describes the proposed DyHF in detail. The experimental results are given in Section V. More discussions on the performance of DyHF are provided in Section VI. Finally, Section VII concludes this paper.

II. Differential Evolution (DE)

DE, which is proposed by Storn and Price [6], implements mutation, crossover, and selection operations to update the population during the evolution. There are several versions for DE. The one used in this paper is called $DE/rand/1/bin$.

The population of DE consists of NP n-dimensional real-valued vectors

$$\vec{x}_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,n}\}, \quad i = 1, 2, \ldots, NP. \tag{6}$$

Taking into account each individual $\vec{x}_i$ (called a target vector), a mutant vector $\vec{v}_i = \{v_{i,1}, v_{i,2}, \ldots, v_{i,n}\}$ is obtained by adding the weighted difference of two population members to a third individual

$$\vec{v}_i = \vec{x}_p + F \cdot (\vec{x}_q - \vec{x}_r) \tag{7}$$

where p, q, and r are randomly selected from $[1, NP]$ and satisfy: $p \neq q \neq r \neq i$, and F is the scaling factor.

After the mutation operation, the trial vector $\vec{u}_i$ is generated by making use of a binomial crossover operation on the target vector $\vec{x}_i$ and the mutant vector $\vec{v}_i$

$$u_{i,j} = \begin{cases} v_{i,j} & \text{if } rand_j \leq C_r \quad or \quad j = j_{rand} \\ x_{i,j} & \text{otherwise.} \end{cases} \tag{8}$$

where $i = 1, 2, \ldots, NP$, $j = 1, 2, \ldots, n$, j_{rand} is a randomly chosen integer between 1 and n, $rand_j$ is the jth evaluation of a uniform random number generator between 0 and 1, and C_r is the crossover control parameter. "$j = j_{rand}$" can guarantee that the trial vector $\vec{u}_i$ differs from its target vector $\vec{x}_i$.

Selection operation is conducted by comparing the target vector $\vec{x}_i$ against the trial vector $\vec{u}_i$. The better one will enter the next generation

$$\vec{x}_i = \begin{cases} \vec{u}_i & \text{if } f(\vec{u}_i) \leq f(\vec{x}_i) \\ \vec{x}_i & \text{otherwise.} \end{cases} \tag{9}$$

III. Related Work

In the special session on constrained real-parameter optimization of the 2006 IEEE congress on evolutionary computation (CEC2006) [7], a variety of COEAs have been proposed. Takahama and Sakai [8] proposed an approach called εDE, which combines the ε constrained method [9] with DE. Liang and Suganthan [10] combined dynamic multiswarm particle swarm optimization (PSO) with a novel constraint-handling mechanism. Brest *et al.* [11] adopted self-adaptive jDE-2 algorithm [12] to solve COPs. In [13] and [14], Zielinski and Laur exploited DE and PSO as the search engine to solve COPs, respectively. Tasgetiren and Suganthan [15] presented a multipopulated DE. A novel method called PESO+ (particle evolutionary swarm optimization plus) is proposed by Munoz-Zavala *et al.* [16]. Mezura-Montes *et al.* [17] presented a modified DE-based approach. This method allows each solution to generate more than one offspring based on a modified DE which combines the information of the best solution and the current solution to define new search directions. Huang *et al.* [18] also introduced a self-adaptive DE for COPs. Runarsson [19] proposed an approximation of evolution strategy (ES) using stochastic ranking [20], the purpose of which is to reduce the number of fitness evaluations. Generalized DE is proposed by Kukkonen and Lampinen [21]. Sinha *et al.* [22] employed a population-based steady-state optimization algorithm for COPs.

During the past four years, solving COPs by EAs has still attracted wide research interest, and some methods have been proposed. Cagnina *et al.* [23] introduced an enhanced PSO algorithm called CPSO-shake. Leguizamón and Coello Coello [24] presented a boundary search-based ant colony optimization algorithm. Mezura-Montes and Cecilia-López-Ramírez [25] presented a comparison of four bio-inspired algorithms (i.e., DE, genetic algorithm, ES, and PSO) to solve 24 benchmark test functions. Huang *et al.* [26] proposed a co-evolutionary DE which employs the notion of co-evolution to adapt penalty factors. Mallipeddi and Suganthan [27] presented empirical studies to evaluate the performance of four constraint-handling methods using an adaptive evolutionary programming. Wang *et al.* [28] proposed an adaptive tradeoff model to handle constraints. Barkat Ullah *et al.* [29] proposed an agent-based memetic algorithm. Huang *et al.* [30] proposed a constrained sorting method which is based on a dynamic penalty function and a nondominated sorting technique. Li *et al.* [31] proposed a DE with level comparison for constrained optimization. Tessema and Yen [32] introduced an adaptive penalty formulation to construct the fitness function. Jia *et al.* [33] proposed a biobjective-based EA called BIEA.

From the above literature review, we can observe that the hybrid evolutionary framework including both the global and local search has seldom been investigated by the researchers in the community of constrained evolutionary optimization. Although HCOEA is an attempt to help close the research gap in this particular topic, the performance of HCOEA is sensitive to some problem-dependent parameters and is not well for some complex COPs. Moreover, the method describing how to make the hybrid evolutionary framework more effective and efficient is usually neglected in HCOEA. These considerations motivate us to present a *dynamic hybrid framework* (DyHF) to deal with COPs.

IV. Proposed Method

In principle, the proposed DyHF belongs to the category of the methods based on multiobjective optimization concepts, since it transforms a COP into a biobjective optimization problem $\vec{f}(\vec{x}) = (f(\vec{x}), G(\vec{x}))$ by treating the degree of constraint violation $G(\vec{x})$ as an additional objective. As a result, the original objective function $f(\vec{x})$ and the degree of constraint violation $G(\vec{x})$ should be considered simultaneously when comparing the individuals in the population. Therefore, Pareto dominance usually used in multiobjective optimization is adopted to compare the individuals in the population. Let $\vec{x}_1$ and $\vec{x}_2$ be two individuals in the population, $\vec{x}_1$ is said to Pareto dominate $\vec{x}_2$ (denoted as $\vec{x}_1 \prec \vec{x}_2$), if 1) $f(\vec{x}_1) \leq f(\vec{x}_2) \wedge G(\vec{x}_1) \leq G(\vec{x}_2)$, and 2) $f(\vec{x}_1) < f(\vec{x}_2) \vee G(\vec{x}_1) < G(\vec{x}_2)$. $\vec{x}_1$ and $\vec{x}_2$ are considered nondominated with each other if they cannot Pareto dominate each other. In addition, $\vec{x}$ is called a nondominated individual in the population if there is no other $\vec{x}'$ in the population such that $\vec{x}' \prec \vec{x}$.

A. Algorithmic Framework

At each generation, DyHF maintains:

1) a population P of NP individuals, i.e., $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_{NP}$;
2) their objective function values $f(\vec{x}_1), f(\vec{x}_2), \ldots, f(\vec{x}_{NP})$ and their degree of constraint violations $G(\vec{x}_1), G(\vec{x}_2), \ldots, G(\vec{x}_{NP})$.

DyHF adopts the following framework for solving COPs:

Step 1. Set $G = 0$. Generate an initial population P by uniformly and randomly sampling from the search space S, evaluate the f-value and the G-value for each individual, and compute the number of feasible solutions (NF) in P.

Step 2. If $rand < (NP - NF)/NP$, where $rand$ is a uniformly distributed random number between 0 and 1, then the local search is applied, otherwise, the global search is applied.

Step 3. Compute the number of feasible solutions (NF) in P.

Step 4. Set $G = G + 1$.

Step 5. If the stopping criterion is met, stop and output the best solution $\vec{x}_{best}$ in P, else go to Step 2.

Next, we will discuss the implementation of the global and local search models in detail.

B. Global Search Model

During the evolution, the global search model mainly focuses on refining the overall performance of the population and exploring more promising region, which works as follows.

Step 1. Each target vector $\vec{x}_i$ $(i = 1, 2, \ldots, NP)$ in the population P is used to create a trial vector $\vec{u}_i$ through the mutation and crossover operations of DE.

Step 2. Compute the f-value and the G-value for the trial vector $\vec{u}_i$.

Step 3. If $\vec{u}_i \prec \vec{x}_i$, the trial vector $\vec{u}_i$ will replace the target vector $\vec{x}_i$, else no replacement occurs.

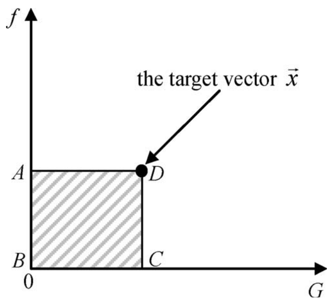

Fig. 1. Schematic diagram to illustrate the Pareto dominance relationship between the target vector and the trial vector.

Since DyHF converts a COP into a biobjective optimization problem $\vec{f}(\vec{x})$, instead of utilizing the comparison criterion in equation (9), the comparison of individuals is based on Pareto dominance in the global search model. By using the trial vector to eliminate the inferior target vector, the update of the population P is achieved.

C. Local Search Model

Although the global search model has the capability to guide the population toward more promising region by replacing the target vector with the superior trial vector, the convergence speed of it may be slow. For instance, if the feasible region occupies a very small part of the whole search space, in the early stage the population may often contain infeasible solutions only. Under this condition, the trial vector may be infeasible and may be frequently nondominated with or dominated by its target vector in the global search model, if we only use Pareto dominance as the comparison criterion. Thus, the population cannot quickly approach the feasible region and may stagnate in the infeasible region. An illustrative example is shown in Fig. 1. In Fig. 1, the target vector $\vec{x}$ is an infeasible solution since $G(\vec{x}) > 0$, the corresponding trial vector $\vec{u}$ can Pareto dominate $\vec{x}$ only when its image in the objective and constraint spaces is located in the region ABCDA. As a result, $\vec{u}$ might be always nondominated with or dominated by $\vec{x}$ and, consequently, $\vec{u}$ cannot survive into the next population.

To address the above issue, the local search model is introduced, the property of which is to efficiently guide the population toward the feasible region from different directions.

In the local search model, the population P is firstly clustered into a number of subpopulations with size NS as follows.

Step 1. Randomly select a reference point $\vec{r}$ from the search space.

Step 2. Find the nearest individual $\vec{x}' \in P$ to $\vec{r}$, which means that compared with the other individuals in P, $\vec{x}'$ has the minimum Euclidean distance from $\vec{r}$. Note that in this paper, Euclidean distance is calculated in the decision space.

Step 3. Combine $NS - 1$ individuals of the population P, which are nearest to $\vec{x}'$, with $\vec{x}'$ to form a subpopulation.

Step 4. Eliminate these NS individuals from P.

Step 5. Execute Step 2–Step 4 continuously until the population P is divided into $\lfloor NP/NS \rfloor$ subpopulations.

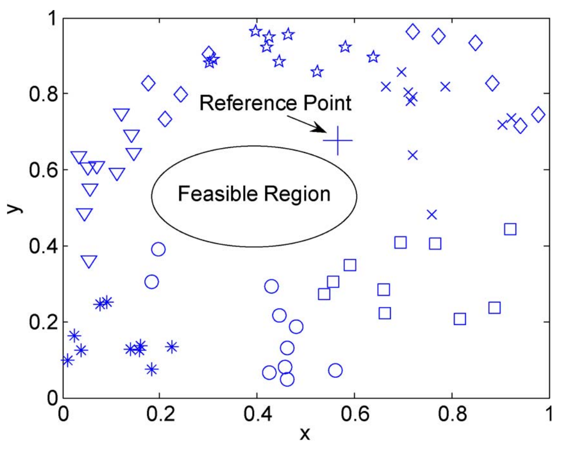

Fig. 2. Two-dimensional search space, the feasible region, the reference point, and the seven subpopulations.

The schematic graph of the above procedure in a 2-D search space is shown in Fig. 2, in which the population P contains 70 individuals and is split to seven subpopulations. Afterward, each subpopulation is evolved according to the following steps.

Step 6. Use the mutation and crossover operations of DE to produce the offspring subpopulation, i.e., the set of all the trial vectors of each subpopulation.

Step 7. The nondominated individuals of the offspring subpopulation are identified, and each nondominated individual is used to replace a randomly selected dominated individual (if it exists) in the subpopulation.

Step 8. Use the best infeasible individual (i.e., the infeasible solution with the lowest degree of constraint violation) of the nondominated individuals in the offspring subpopulation to randomly replace an individual in the subpopulation, if all the nondominated individuals in the offspring subpopulation are infeasible, and this best infeasible individual does not replace any individual in the subpopulation from Step 7.

The main reason why Step 7 is applied is that the nondominated individuals represent the most important information of the population to which they belong [4]. The aim of Step 8 is to motivate each subpopulation toward the feasible region constantly. By partitioning the population into different subpopulations, the local search model is intended to approach the feasible region from different directions.

D. Dynamic Implementation of the Global and Local Search Models

As pointed out previously, the local search model is mainly exploited to motivate the population to approach or enter the feasible region from different directions promptly, since there is no prior knowledge about the location and shape of the feasible region. Therefore, the local search model is very useful for the search in the early stage. However, the primary purpose of the global search model is to sample more promising region. Indeed, the global search model can play an important role only when the population has approached or entered the feasible region, since if the population is very far away from the feasible region, the behavior of the global search model is similar to random walk in the search space (as analyzed in Section IV-C). In particular, if the global search model has concrete capability to explore the feasible region under the condition that the population is close to or enters the feasible region, the algorithm is able to find the global optimum with a higher probability. Meanwhile, it is noteworthy that if the local search model is conducted too frequently when the population has approached or entered the feasible region, the computational resource might be wasted seriously.

Based on the above consideration, it is more reasonable to dynamically implement the local and global search models according to the state of the population to support both convergence and feasibility. In this paper, the application of the global and local search models is dynamically determined according to the feasibility proportion of the current population. By doing this, we attempt to control the frequency of applying the global search and the local search to maintain a balance between searching for feasible solutions and converging to fitter solutions. In the early stage, the feasibility proportion of the population may be very small or equal to zero. In this case, the local search model is performed with a higher frequency to guide the population toward feasibility promptly. Along with the evolution, the population may converge near the feasible region, and some individuals of the population may be feasible. Thus, the global search is gradually strengthened to encourage the individuals to probe more promising region (especially to probe the feasible region). At the later stage of search, most of the individuals of the population may be feasible, and, as a result, the local search model is executed with a less probability, and the global search model is mainly used to refine the overall quality of the population. Hence, the above dynamic implementation of the global and local search models in DyHF can switch smoothly from feasibility to convergence.

Schematic diagrams of the global and local search models are depicted in Fig. 3. From Fig. 3(a), we can observe that the feasible region is very small compared with the whole search space, and the global optimum is located on the boundary of the feasible region. Suppose that the initial population contains nine individuals randomly chosen from the search space [Fig. 3(a)]. Since there is no feasible solution in the initial population, only the local search model will be implemented in DyHF. In the local search model, the population will be divided into several subpopulations, and each subpopulation has its search direction to approach the feasible region [Fig. 3(b)]. During the evolution, some individuals will enter the feasible solution and become feasible [Fig. 3(c)]. Under this condition, if we still only apply the local search model, the population will be partitioned into several subpopulations [Fig. 3(d)], and the search ability of each subpopulation might be weak in the feasible region. As a result, each subpopulation will converge to a local optimal basin, and the global optimum cannot be attained in the end [Fig. 3(e)]. However, in DyHF, the global search model will be incorporated under the condition shown in Fig. 3(c). By making use of the global search model, the search of the population will be diversified, and the global exploration ability of the population will be enhanced significantly. As a result, the population can gradually converge to the global optimum [Fig. 3(f) and (g)].

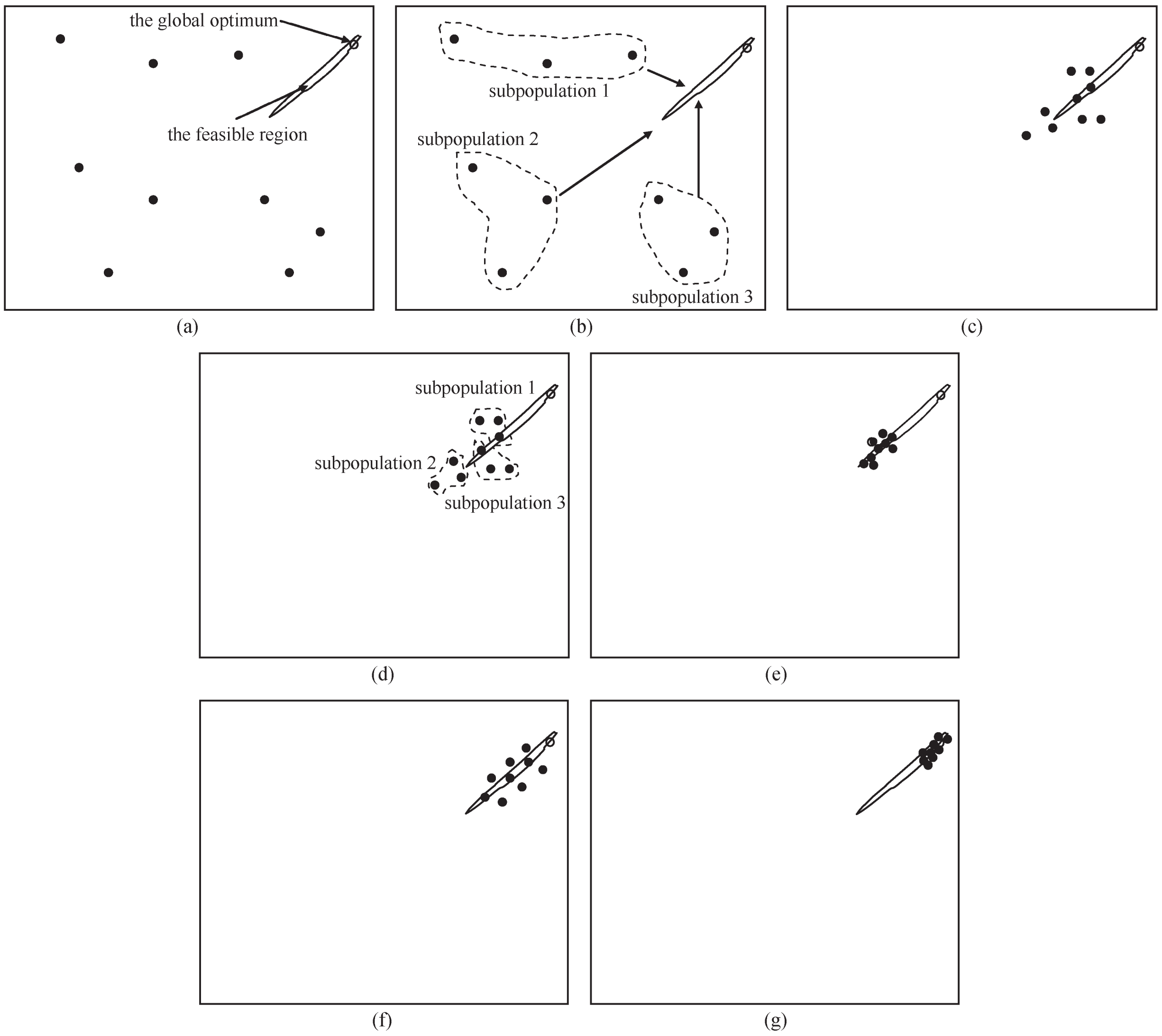

Fig. 3. Schematic diagrams to illustrate the global search model and the local search model.

Remark 1: The feasibility proportion of the population has also been exploited by Wang *et al.* [28] and Tessema and Yen [32]; however, in these two papers the feasibility proportion is used to construct the fitness function, by which some potential feasible and infeasible solutions can survive into the next population. In contrast, in this paper, the feasibility proportion is used to control the frequency of the implementation of the global and local search models. Therefore, the idea of this paper is quite different from that of [28] and [32].

Remark 2: Ray and Liew [34] proposed a society and civilization model to solve COPs. In this model, a society contains a cluster of individuals in the search space, and a civilization is a set of such societies. During the evolution, the quality of the individuals in a society is enhanced by learning from the leaders of this society (called intrasociety information exchange). In addition, the leaders of a society improve their quality by learning from the leaders of other societies (called intersociety information exchange). Indeed, the intersociety information exchange and the intrasociety information exchange are similar to the global search and the local search, respectively. However, the principle and motivation of the global and local search in DyHF are quite different from those of the intersociety and intrasociety information exchange in [34]. Moreover, in this paper, the global search and the local search are executed dynamically.

V. Experimental Results

A. Experimental Settings

Although 24 benchmark test functions are collected in [7], we do not use two of them to evaluate DyHF because almost no approaches can find the feasible solutions for these two test functions (i.e., g20 and g22) so far. Information on these test functions is tabulated in [7]. It is necessary to notice that an improved best known solution has been found in this paper for test function g17, which is $\vec{x}^* = (201.784462493550, 100.000000000000, 383.071034852773, 419.999999999999, -10.907793031517, 0.073148231208)$ with $f(\vec{x}^*) = 8853.533874806482$. The similar result has also been reported in [21] for this test function.

DyHF includes six parameters: the size of the population P (NP), the size of each subpopulation (NS), the scaling factor and the crossover control parameter of DE in the local search model (F_1 and C_{r1}) and in the global search model (F_2 and C_{r2}).

TABLE I
FUNCTION ERROR VALUES ACHIEVED WHEN FES $= 5 \times 10^3$, FES $= 5 \times 10^4$, AND FES $= 5 \times 10^5$ FOR TEST FUNCTIONS G01-G06

FES	Prob.	g01	g02	g03	g04	g05	g06
5×10^3	Best	7.2841E+00 (0)	3.7709E-01 (0)	6.0798E-01 (0)	4.5215E+01 (0)	9.3455E+00 (3)	3.2386E+00 (0)
	Median	9.2118E+00 (0)	4.5073E-01 (0)	6.1934E-01 (1)	1.3730E+02 (0)	-1.0673E+01 (3)	7.4560E+01 (0)
	Worst	1.2101E+01 (1)	5.0184E-01 (0)	9.6670E-01 (1)	3.7077E+02 (0)	5.9949E+00 (3)	4.3250E+02 (0)
	c	0, 0, 0	0, 0, 0	0, 0, 1	0, 0, 0	3, 0, 0	0, 0, 0
	Mean	9.4246E+00	4.4517E-01	9.6967E-01	1.6291E+02	2.1005E+02	9.4927E+01
	Std	1.4206E+00	3.2398E-02	5.8467E-02	7.2031E+01	2.9299E+02	9.0041E+01
5×10^4	Best	2.0857E-03 (0)	4.2266E-02 (0)	1.1117E-05 (0)	7.5828E-06 (0)	5.0667E-06 (0)	8.7184E-09 (0)
	Median	4.0385E-03 (0)	8.6196E-02 (0)	8.2363E-05 (0)	5.8666E-05 (0)	2.0627E-05 (0)	1.0582E-06 (0)
	Worst	8.8146E-03 (0)	1.3648E-01 (0)	1.9301E-04 (0)	7.9138E-05 (0)	1.4635E-04 (0)	8.3363E-05 (0)
	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	Mean	4.5037E-03	9.0115E-02	9.7628E-05	6.2016E-05	4.1867E-05	9.8924E-06
	Std	2.0179E-03	2.6079E-02	9.1654E-05	3.4439E-05	8.8367E-05	1.9976E-05
5×10^5	Best	0.0000E+00 (0)	7.6740E-10 (0)	-2.8865E-15 (0)	-3.6379E-12 (0)	-1.8189E-12 (0)	-1.6370E-11 (0)
	Median	0.0000E+00 (0)	1.1089E-08 (0)	-2.8865E-15 (0)	-3.6379E-12 (0)	-1.8189E-12 (0)	-1.6370E-11 (0)
	Worst	0.0000E+00 (0)	6.6810E-08 (0)	-2.4424E-15 (0)	-3.6379E-12 (0)	-1.8189E-12 (0)	-1.6370E-11 (0)
	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	Mean	0.0000E+00	1.5862E-08	-2.7711E-15	-3.6379E-12	-1.8189E-12	-1.6370E-11
	Std	0.0000E+00	1.5151E-08	1.5857E-16	0.0000E+00	0.0000E+00	0.0000E+00

There are some guidelines for determining the settings of the aforementioned six parameters. To avoid premature convergence, the size of the population P should be relatively large. As analyzed in [4], to obtain competitive performance, a bigger value for the size of the subpopulation is needed. There are two reasons: 1) the subpopulation with a bigger size is more powerful to probe a larger search region and to rapidly approach the feasible region than the subpopulation with a smaller size; and 2) the subpopulation of a bigger size can involve both feasible and infeasible solutions with a higher probability than the subpopulation of a smaller size. It is important to note that applying crossover and mutation operations of DE to both feasible and infeasible solutions in some promising region is very useful for the algorithm to find the optimal solution exactly located on the boundary of the feasible region.

In addition, the choice of the control parameters in DE is also critical for the performance of DyHF. Since the local search model of DyHF aims at promptly guiding the subpopulations toward the feasible region from various directions, each subpopulation should be capable of exploring more promising regions constantly, and the diversity of each subpopulation should be maintained. As a result, relatively bigger values are required for F_1 and C_{r1}. The bigger value of F_1 means that the mutant vectors generated by the mutation operation are distributed widely and that the mutation is able to sufficiently sample the region of interest. In addition, the bigger value of C_{r1} means that the offspring population generated via the crossover operation is quite different from the parent population, since the offspring population will inherit little information from the parent population. Thus, the diversity of the subpopulation can be kept. In addition, as analyzed previously, the global search model plays its important role mainly in the middle and later stages of the evolution; therefore, the value of F_2 should be slightly smaller so as to speed up the convergence of the algorithm.

Based on the above discussion, the actual parameter settings are $NP = 140$, $NS = 10$, $F_1 = 0.7$, $C_{r1} = 1.0$, $F_2 = 0.5$, and C_{r2} is set as follows:

$$C_{r2} = \begin{cases} 1.0 & \text{if } rand < P_{c_{r2}} \\ 0.1 & \text{otherwise} \end{cases} \tag{10}$$

where $rand$ is a uniformly distributed random number between 0 and 1. The main motivation of equation (10) is to balance the exploration and exploitation of the population in the global search model. If $C_{r2} = 1.0$, no information of the offspring is derived from the target vector, and, as a result, the population focuses on exploration. On the other hand, if $C_{r2} = 0.1$, the offspring gets more information from the target vector, and the search focuses on the neighborhood of the current population, which can accelerate the convergence of the population. We also observe that if "$C_{r2} = 1.0$" is used with a slightly higher probability than "$C_{r2} = 0.1$," the performance of DyHF will be more competitive. Hence, $P_{c_{r2}}$ is set to 0.75 in this paper, which encourages more exploration characteristic.

B. Experimental Results

The performance of DyHF is assessed on 22 benchmark test functions. For each test function, 25 independent runs are performed using 5×10^5 fitness evaluations (FES) at maximum. The tolerance value δ for the equality constraints is set to 0.0001 according to the suggestion in [7].

Tables I–IV record the best, median, worst, mean, and standard deviation of the function error value $(f(\vec{x}) - f(\vec{x}^*))$ for the achieved best solution $\vec{x}$ after 5×10^3 FES, 5×10^4 FES, and 5×10^5 FES for each test function. In these tables, numbers in the parenthesis after the function error values of the best, median, and worst solutions show the corresponding number of violated constraints at the best, median, and

TABLE II
FUNCTION ERROR VALUES ACHIEVED WHEN FES $= 5 \times 10^3$, FES $= 5 \times 10^4$, AND FES $= 5 \times 10^5$ FOR TEST FUNCTIONS G07-G12

FES \ Prob.		g07	g08	g09	g10	g11	g12
5×10^3	Best	3.9056E+01 (0)	1.9667E-08 (0)	7.4724E+00 (0)	4.2652E+03 (0)	2.0256E-05 (0)	2.6445E-06 (0)
	Median	1.2829E+02 (0)	2.6179E-08 (0)	6.2201E+01 (0)	9.0814E+03 (0)	1.4930E-04 (0)	3.6353E-05 (0)
	Worst	4.1230E+02 (0)	3.4798E-08 (0)	1.0088E+02 (0)	9.4713E+03 (1)	7.1381E-04 (0)	2.1788E-04 (0)
	c	1, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	Mean	1.3768E+02	2.5249E-08	6.1185E+01	8.9299E+03	2.0494E-04	5.2996E-05
	Std	8.5315E+01	9.2952E-09	2.5219E+01	3.4972E+03	1.7648E-04	4.9497E-05
5×10^4	Best	1.5531E-02 (0)	2.7756E-17 (0)	4.1724E-07 (0)	1.6261E+01 (0)	3.4416E-15 (0)	0.0000E+00 (0)
	Median	3.7607E-02 (0)	4.1633E-17 (0)	3.1150E-06 (0)	3.9875E+01 (0)	3.4661E-13 (0)	0.0000E+00 (0)
	Worst	9.9396E-02 (0)	4.1633E-17 (0)	2.0542E-05 (0)	7.8612E+01 (0)	8.8889E-11 (0)	0.0000E+00 (0)
	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	Mean	3.9911E-02	3.6082E-17	3.9154E-06	4.4063E+01	7.5562E-12	0.0000E+00
	Std	1.6690E-02	6.9388E-18	4.4115-06	1.6914E+01	2.2372E-11	0.0000E+00
5×10^5	Best	-2.3803E-13 (0)	2.7756E-17 (0)	-2.2737E-13 (0)	-7.2760E-12 (0)	0.0000E+00 (0)	0.0000E+00 (0)
	Median	-2.2382E-13 (0)	2.7756E-17 (0)	-2.2737E-13 (0)	-7.2759E-12 (0)	0.0000E+00 (0)	0.0000E+00 (0)
	Worst	-2.2026E-13 (0)	2.7756E-17 (0)	-2.2737E-13 (0)	-6.3664E-12 (0)	0.0000E+00 (0)	0.0000E+00 (0)
	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	Mean	-2.2524E-13	2.7756E-17	-2.2737E-13	-7.1668E-12	0.0000E+00	0.0000E+00
	Std	4.4704E-15	0.0000E+00	0.0000E+00	3.0164E-13	0.0000E+00	0.0000E+00

TABLE III
FUNCTION ERROR VALUES ACHIEVED WHEN FES $= 5 \times 10^3$, FES $= 5 \times 10^4$, AND FES $= 5 \times 10^5$ FOR TEST FUNCTIONS G13-G18

FES \ Prob.		g13	g14	g15	g16	g17	g18
5×10^3	Best	4.2206E-02 (3)	-3.3392E+01 (3)	7.3492E-02 (2)	6.0594E-02 (0)	3.5142E+02 (4)	5.4452E+00 (4)
	Median	2.1162E-01 (3)	-6.6941E+01 (3)	2.5700E-01 (2)	1.3785E-01 (0)	1.9451E+02 (4)	1.0559E+00 (5)
	Worst	9.4092E-01 (3)	-1.6869E+02 (3)	1.6092E-02 (2)	2.7384E-01 (0)	-6.0225E+01 (4)	-4.5079E-01 (8)
	c	0, 3, 0	3, 0, 0	0, 2, 0	0, 0, 0	4, 0, 0	2, 3, 0
	Mean	5.7824E-01	-7.4442E+01	2.4183E-01	1.4394E-01	7.1915E+01	4.1775E-01
	Std	3.8798E-01	3.1511E+01	3.4371E-01	5.5245E-02	2.3229E+02	9.0234E-01
5×10^4	Best	4.8305E-09 (0)	1.6873E-03 (0)	2.8649E-10 (0)	2.5751E-06 (0)	1.2834E+00 (0)	1.0491E-03 (0)
	Median	1.0208E-07 (0)	3.1789E-03 (0)	4.3398E-09 (0)	8.5111E-06 (0)	6.9244E+00 (0)	4.5668E-03 (0)
	Worst	5.0902E-03 (0)	8.8342E-03 (0)	2.8980E-08 (0)	2.4892E-05 (0)	7.6495E+01 (0)	8.8302E-03 (0)
	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	Mean	2.3131E-04	3.9047E-03	7.8852E-09	9.7633E-06	1.0367E+01	4.7255E-03
	Std	1.0162E-03	1.8256E-03	8.3663E-09	5.7313E-06	1.4743E+01	2.2390E-03
5×10^5	Best	-2.2204E-16 (0)	1.4210E-14 (0)	-1.1368E-13 (0)	3.7748E-15 (0)	0.0000E+00 (0)	2.2204E-16 (0)
	Median	-1.9428E-16 (0)	1.4210E-14 (0)	-1.1368E-13 (0)	3.7748E-15 (0)	0.0000E+00 (0)	2.2204E-16 (0)
	Worst	-1.9428E-16 (0)	1.4210E-14 (0)	-1.1368E-13 (0)	3.7748E-15 (0)	1.8190E-12 (0)	2.2204E-16 (0)
	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	Mean	-2.0761E-16	1.4210E-14	-1.1368E-13	3.7748E-15	7.2759E-13	2.2204E-16
	Std	1.4152E-17	0.0000E+00	0.0000E+00	0.0000E+00	9.5099E-13	0.0000E+00

worst solutions, respectively. Numbers of constraints, the violations of which at the median solution are within the intervals $[1.0, \infty)$, [0.01, 1.0) and [0.0001, 0.01), are shown in c, respectively.

One of the first observations from Tables I–IV is that for all the test functions, except for test function g21, feasible solutions can be consistently found using 5×10^4 FES. For test function g21, feasible solutions can be found in all trials using 5×10^5 FES.

From Tables I–IV, we also observe that DyHF can produce the results relatively near the best known solutions by using 5×10^4 FES for test functions g04, g06, g08, g09, g11, g12, g15, g16, and g24. Moreover, the function error values provided by DyHF are extremely small for all the test functions in 5×10^5 FES. The above behavior exhibits the great robustness of DyHF.

Table V shows the number of FES to achieve the success condition: $f(\vec{x}) - f(\vec{x}^*) \leq 0.0001$ and $\vec{x}$ is feasible, the feasible rate (the percentage of runs where at least one feasible solution is found), the success rate (the percentage of runs where the algorithm finds a solution that satisfies the success condition), and the success performance (the mean number of FES for successful runs multiplied by the number of total runs and divided by the number of successful runs) for all the 22 test functions.

It can be observed from Table V that DyHF achieves 100% feasible rate and 100% success rate for all the 22 test functions, which means DyHF has the capability to solve these test functions consistently. From Table V, it is also obvious that DyHF uses less than 5×10^4 FES for 12 test functions, less than 1.5×10^5 FES for 20 test functions, and less than 2.2×10^5 FES for all the test functions, to achieve the success condition under the mean condition.

TABLE IV
FUNCTION ERROR VALUES ACHIEVED WHEN FES $= 5 \times 10^3$, FES $= 5 \times 10^4$, AND FES $= 5 \times 10^5$ FOR TEST FUNCTIONS G19, G21, G23, AND G24

FES \ Prob.		g19	g21	g23	g24
5×10^3	Best	1.8629E+02 (0)	1.1590E+02 (5)	-2.2944E+02 (5)	1.9054E-03 (0)
	Median	2.6765E+02 (0)	1.6914E+02 (5)	-1.8709E+02 (4)	1.4943E-02 (0)
	Worst	4.7798E+02 (0)	2.2934E+01 (5)	-2.9325E+02 (4)	2.5882E-02 (0)
	c	0, 0, 0	1, 4, 0	3, 1, 0	0, 0, 0
	Mean	2.8933E+02	1.6414E+02	-2.9184E+02	1.3918E-02
	Std	7.8347E+01	1.6745E+02	4.1202E+02	7.1598E-03
5×10^4	Best	2.6014E-01 (0)	2.4190E-02 (0)	7.2720E+00 (0)	3.2862E-14 (0)
	Median	5.1157E-01 (0)	7.1449E-01 (0)	2.9353E+01 (0)	7.1054E-13 (0)
	Worst	8.4759E-01 (0)	1.2779E+02 (4)	1.6454E+02 (0)	7.8923E-12 (0)
	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	Mean	5.1185E-01	2.8615E+01	4.1483E+01	6.5881E-13
	Std	1.6976E-01	5.1238E+01	3.4086E+01	1.6951E-12
5×10^5	Best	2.1316E-14 (0)	-4.1487E-10 (0)	-6.8212E-13 (0)	3.2862E-14 (0)
	Median	2.8421E-14 (0)	-3.0399E-10 (0)	-4.5474E-13 (0)	3.2862E-14 (0)
	Worst	2.8421E-14 (0)	-2.2754E-10 (0)	-1.1368E-13 (0)	3.2862E-14 (0)
	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	Mean	2.6716E-14	-3.0739E-10	-4.7293E-13	3.2862E-14
	Std	3.0971E-15	5.1155E-11	1.3403E-13	0.0000E+00

TABLE V
NUMBER OF FES TO ACHIEVE THE SUCCESS CONDITION, SUCCESS RATE, FEASIBLE RATE, AND SUCCESS PERFORMANCE

Prob.	Best	Median	Worst	Mean	Std	Feasible Rate	Success Rate	Success Performance
g01	62720	68880	74760	69098	2670.4	100%	100%	69098
g02	91280	110040	141400	111428	12945.3	100%	100%	111428
g03	34860	42560	51800	42943	4396.5	100%	100%	42943
g04	34300	40880	45920	40235	3480.7	100%	100%	40235
g05	41580	46480	54600	47236	3914.9	100%	100%	47236
g06	27580	37240	49700	37720	5057.8	100%	100%	37720
g07	85540	93380	102620	94150	4352.7	100%	100%	94150
g08	840	1120	1400	1223	283.1	100%	100%	1223
g09	36680	41300	47600	41406	2536.8	100%	100%	41406
g10	121380	144060	163380	142652	12025.7	100%	100%	142652
g11	3500	5600	8960	5768	1289.4	100%	100%	5768
g12	420	2940	5600	3012	1371.1	100%	100%	3012
g13	21000	26600	133000	32478	22789.6	100%	100%	32478
g14	57960	64540	68740	64579	2876.1	100%	100%	64579
g15	15540	22680	32480	23038	4325.8	100%	100%	23038
g16	25620	31080	36960	30300	2770.0	100%	100%	30300
g17	83440	178360	452620	210212	95867.8	100%	100%	210212
g18	76300	87780	101780	88379	6689.9	100%	100%	88379
g19	108780	117740	128380	114933	5934.1	100%	100%	114933
g21	68180	91560	235900	102180	34905.7	100%	100%	102180
g23	134260	154560	229040	161145	20907.8	100%	100%	161145
g24	11620	14700	18340	14284	1809.1	100%	100%	14284

C. Comparison With Some State-of-the-Art Approaches in Constrained Evolutionary Optimization

This subsection presents a performance comparison between DyHF and six state-of-the-art approaches: εDE [8], jDE-2 [11], MPDE [15], MDE [17], SaDE [18], and GDE [21]. It is necessary to note that DE is adopted as the search engine by the above six methods and DyHF. The comparative results are presented in Table VI in terms of two performance indicators, i.e., feasible rate and success rate. The experimental results of the above six methods are directly taken from the literature. Note that the experimental results of those test functions for which all the methods compared can achieve both 100% feasible rate and 100% success rate are omitted due to space limitations.

As shown in Table VI, DyHF has similar performance with εDE, jDE-2, MDE, and SaDE and exhibits superior performance compared with MPDE and GDE, in terms of the mean feasible rate. Although the mean feasible rates of εDE and SaDE are outstanding, it is necessary to note that gradient information of test functions is required by these two methods.

In terms of the success rate, no performance difference is found between εDE and DyHF. However, it is noteworthy that DyHF is based on pure evolutionary methods compared with εDE. In addition, the mean success rate of DyHF is significantly better than that of jDE-2, MPDE, MDE, SaDE, and GDE. Furthermore, an improved best known solution has been found

TABLE VI
COMPARISON OF DYHF WITH RESPECT TO εDE [8], JDE-2 [11], MPDE [15], MDE [17], SADE [18], AND GDE [21], IN TERMS OF FEASIBLE RATE AND SUCCESS RATE

Prob.	Feasible rate							Success rate						
	εDE	jDE-2	MPDE	MDE	SaDE	GDE	DyHF	εDE	jDE-2	MPDE	MDE	SaDE	GDE	DyHF
g02	100%	100%	100%	100%	100%	100%	100%	100%	92%	92%	16%	84%	72%	100%
g03	100%	100%	100%	100%	100%	96%	100%	100%	0%	84%	100%	96%	4%	100%
g05	100%	100%	100%	100%	100%	96%	100%	100%	68%	100%	100%	100%	92%	100%
g11	100%	100%	100%	100%	100%	100%	100%	100%	96%	96%	100%	100%	100%	100%
g13	100%	100%	88%	100%	100%	88%	100%	100%	0%	48%	100%	100%	40%	100%
g14	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	80%	96%	100%
g15	100%	100%	100%	100%	100%	100%	100%	100%	96%	100%	100%	100%	96%	100%
g17	100%	100%	96%	100%	100%	76%	100%	100%	4%	28%	100%	4%	16%	100%
g18	100%	100%	100%	100%	100%	84%	100%	100%	100%	100%	100%	92%	76%	100%
g19	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	0%	100%	88%	100%
g21	100%	100%	100%	100%	100%	88%	100%	100%	92%	68%	100%	60%	60%	100%
g23	100%	100%	100%	100%	100%	88%	100%	100%	92%	100%	100%	88%	40%	100%
Mean	100%	100%	99.27%	100%	100%	96.18%	100%	100%	83.64%	91.64%	91.64%	91.09%	80.91%	100%

TABLE VII
COMPARISON OF DYHF WITH RESPECT TO εDE [8], JDE-2 [11], MPDE [15], MDE [17], AND GDE [21], IN TERMS OF THE EFFICIENCY

Prob.	Success performance						Prob.	Success performance					
	εDE	jDE-2	MPDE	MDE	GDE	DyHF		εDE	jDE-2	MPDE	MDE	GDE	DyHF
g01	5.9E+04	5.0E+04	4.3E+04	7.5E+04	4.1E+04	6.9E+04	g12	4.1E+03	6.4E+03	4.2E+03	1.3E+03	3.1E+03	3.0E+03
g02	1.5E+05	1.5E+05	3.0E+05	6.0E+04	1.5E+05	1.1E+05	g13	3.5E+04	*NA*	7.4E+05	2.2E+04	8.7E+05	3.2E+04
g03	8.9E+04	*NA*	2.5E+04	4.5E+04	3.5E+06	4.3E+04	g14	1.1E+05	9.8E+04	4.3E+04	2.9E+05	2.3E+05	6.5E+04
g04	2.6E+04	4.1E+04	2.1E+04	4.2E+04	1.5E+04	4.0E+04	g15	8.4E+04	2.4E+05	2.0E+05	1.0E+04	7.5E+04	2.3E+04
g05	9.7E+04	4.5E+05	2.2E+05	2.1E+04	1.9E+05	4.7E+04	g16	1.3E+04	3.2E+04	1.3E+04	8.7E+03	1.3E+04	3.0E+04
g06	7.4E+03	2.9E+04	1.1E+04	5.2E+03	6.5E+03	3.8E+04	g17	9.9E+04	1.1E+07	7.3E+05	2.6E+04	2.1E+06	2.1E+05
g07	7.4E+04	1.3E+05	5.7E+04	1.9E+05	1.2E+05	9.4E+04	g18	5.9E+04	1.0E+05	4.4E+04	1.0E+05	4.8E+05	8.9E+04
g08	1.1E+03	3.2E+03	1.5E+03	9.2E+02	1.5E+03	1.2E+03	g19	3.5E+05	2.0E+05	1.2E+05	*NA*	2.3E+05	1.1E+05
g09	2.3E+04	5.5E+04	2.1E+04	1.6E+04	3.0E+04	4.1E+04	g21	1.4E+05	1.3E+05	2.1E+05	1.1E+05	5.8E+05	1.0E+05
g10	1.1E+05	1.5E+05	4.8E+04	1.6E+05	8.3E+04	1.4E+05	g23	2.0E+05	3.6E+05	2.1E+05	3.6E+05	1.1E+06	1.6E+05
g11	1.6E+04	5.4E+04	2.3E+04	3.0E+03	8.5E+03	5.8E+03	g24	3.0E+03	1.0E+04	4.3E+03	1.8E+03	3.1E+03	1.4E+04

Sum of success performance	εDE	1.75E+06
	jDE-2	1.33E+07+2**NA*
	MPDE	3.09E+06
	MDE	1.56E+06+*NA*
	GDE	9.83E+06
	DyHF	**1.47E+06**

in this paper for test function g17. Based on this solution, the six methods except for GDE are unable to solve this test function even in one run, according to the target accuracy level. Moreover, GDE obtains this solution only a few times out of 25 runs.

In addition to the feasible rate and the success rate, DyHF has also been compared with these six methods in terms of the efficiency. Indeed, the efficiency of a method can be measured by the success performance. Table VII summarizes the success performance of the different methods, where "NA" denotes that the success performance is not available since the corresponding success rate of a method is 0%. In Table VII, the success performance of εDE, jDE-2, MPDE, MDE, and GDE is directly taken from the literature. Note that for SaDE, the sequential quadratic programming method is exploited as the local search operator; thus, the number of FES exhausted by the local search cannot be exactly computed. Therefore, the experimental results of SaDE have not been included in Table VII. We test the efficiency of these methods by the sum of the success performance, since the evaluation of the objective function and the degree of constraint violation may consume the main computation time at each generation, especially for some complicated COPs. From Table VII, it is clear that DyHF exhibits the highest efficiency as far as the sum of the success performance is concerned.

To conclude, the results obtained in this paper are highly competitive. In addition, our approach is also quite efficient. The above discussion substantially establishes DyHF as a competitive alternative for constrained real-parameter optimization.

VI. DISCUSSION

A comprehensive set of experiments are conducted in this section to verify the effectiveness of some mechanisms proposed in this paper and to explain the rationality behind the parameter settings. To maintain a fair comparison for all experiments: 1) the parameter settings are the same as those recommended in Section V-A, unless we mention new settings for one or some of them to serve the purpose of parameter study; 2) for all conducted experiments, 25 independent trials are executed; 3) only the experimental results of those test functions, for which at least one of the compared methods

TABLE VIII
MEAN OBJECTIVE FUNCTION VALUE, SUCCESS RATE, AND FEASIBLE RATE ON TEST FUNCTIONS G01, G02, G03, G05, G06, G07, G10, G13, G14, G15, G16, G17, G18, G19, G21, AND G23 WITH DIFFERENT PROBABILITIES OF APPLYING THE LOCAL SEARCH MODEL OVER 25 RUNS

Prob.	Mean objective function value (Success rate) [Feasible rate]					
	0.0	0.3	0.5	0.7	1.0	dynamic
g01	-35.9170 (0%) [100%]	-15.0000 (100%) [100%]	-15.0000 (100%) [100%]	-15.0000 (100%) [100%]	-14.8731 (92%) [100%]	-15.0000 (100%) [100%]
g02	-0.8036 (100%) [100%]	-0.8017 (80%) [100%]	-0.7977 (64%) [100%]	-0.7770 (24%) [100%]	-0.5562 (0%) [100%]	-0.8036 (100%) [100%]
g03	-0.7662 (0%) [32%]	-1.0005 (100%) [100%]	-1.0005 (100%) [100%]	-1.0005 (100%) [100%]	-0.9996 (92%) [100%]	-1.0005 (100%) [100%]
g05	4892.751 (0%) [0%]	5126.4967 (100%) [100%]	5126.4967 (100%) [100%]	5126.4967 (100%) [100%]	5126.4991 (76%) [100%]	5126.4967 (100%) [100%]
g06	-6957.3330 (0%) [100%]	-6961.8139 (100%) [100%]	-6961.8139 (100%) [100%]	-6961.8139 (100%) [100%]	-6961.8139 (100%) [100%]	-6961.8139 (100%) [100%]
g07	24.3066 (80%) [100%]	24.3062 (100%) [100%]	24.3062 (100%) [100%]	24.3062 (100%) [100%]	24.3092 (92%) [100%]	24.3062 (100%) [100%]
g10	7291.789 (0%) [0%]	7049.2480 (100%) [100%]	7049.2480 (100%) [100%]	7049.2480 (100%) [100%]	7054.9145 (0%) [100%]	7049.2480 (100%) [100%]
g13	0.0567 (0%) [0%]	0.0539 (100%) [100%]	0.0539 (100%) [100%]	0.0539 (100%) [100%]	0.0540 (84%) [100%]	0.0539 (100%) [100%]
g14	-553.5743 (0%) [0%]	-47.7649 (100%) [100%]	-47.7649 (100%) [100%]	-47.7649 (100%) [100%]	-47.7649 (100%) [100%]	-47.7649 (100%) [100%]
g15	962.4768 (0%) [0%]	961.7150 (100%) [100%]	961.7150 (100%) [100%]	961.7150 (100%) [100%]	961.7150 (100%) [100%]	961.7150 (100%) [100%]
g16	-1.8425 (0%) [72%]	-1.9052 (100%) [100%]	-1.9052 (100%) [100%]	-1.9052 (100%) [100%]	-1.9052 (100%) [100%]	-1.9052 (100%) [100%]
g17	8486.5981 (0%) [0%]	8853.5339 (100%) [100%]	8853.6576 (88%) [100%]	8854.1431 (60%) [100%]	8865.4241 (0%) [100%]	8853.5339 (100%) [100%]
g18	-10.5337 (0%) [0%]	-0.8660 (100%) [100%]	-0.8660 (100%) [100%]	-0.8660 (100%) [100%]	-0.8637 (56%) [100%]	-0.8660 (100%) [100%]
g19	32.6556 (100%) [100%]	32.6556 (100%) [100%]	32.6556 (100%) [100%]	32.6556 (100%) [100%]	33.1331 (0%) [100%]	32.6556 (100%) [100%]
g21	5.5802 (0%) [0%]	214.6811 (84%) [100%]	206.1826 (84%) [100%]	193.7245 (100%) [100%]	216.7448 (4%) [100%]	193.7245 (100%) [100%]
g23	-1971.1829 (0%) [0%]	-340.0533 (80%) [100%]	-388.0547 (96%) [100%]	-400.0551 (100%) [100%]	-372.1099 (0%) [100%]	-400.0551 (100%) [100%]

cannot achieve both 100% success rate and 100% feasible rate, are summarized due to space limitations; and 4) we summarize the experimental results in terms of three indicators, i.e., mean objective function value, success rate, and feasible rate.

A. Effectiveness of Dynamic Implementation of the Global and Local Search Models

Unlike the previous studies, in this paper, the global and local search models are implemented dynamically based on the feasibility proportion of the current population. To verify that the achieved performance of our algorithm is really due to utilizing the dynamic implementation of these two models, all parts of our algorithm are kept untouched, and the local search model is performed with different constant probabilities, i.e., 0.0, 0.3, 0.5, 0.7, and 1.0. The probabilities "0.0" and "1.0" signify that DyHF only includes the global search model and the local search model, respectively. Table VIII summarizes the experimental results of test functions g01, g02, g03, g05, g06, g07, g10, g13, g14, g15, g16, g17, g18, g19, g21, and g23 for different algorithms. Our original algorithm is referred as "dynamic" in Table VIII.

From Table VIII, it is clear that performance degradation arises when implementing DyHF with constant probabilities. If only the global search is adopted (i.e., the probability is equal to 0.0), the algorithm cannot consistently find the feasible solutions for test functions g03, g05, g10, g13, g14, g15, g16, g17, g18, g21, and g23. It is because the global search model cannot guide the population toward the feasible region effectively (as analyzed in Section IV-C). If only the local search is adopted (i.e., the probability is equal to 1.0), although the algorithm is able to enter the feasible region in all runs, it fails to solve test functions g01, g02, g03, g05, g07, g10, g13, g17, g19, g21, and g23 in some trials. It is because the ability of the algorithm to improve the feasible solutions inside the feasible region is not good due to the lack of the global search. When the local search is performed with a relatively small probability (i.e., 0.3), the algorithm is unable to solve test functions g02, g21, and g23 consistently. When the local and global search models are executed with an equal probability (i.e., 0.5), the algorithm is likely to get stuck at a local optimum for test functions g02, g17, g21, and g23. When the local search model is applied with a relatively large probability, (i.e., 0.7), premature convergence arises for test functions g02 and g17. The above phenomena are because DyHF with constant probabilities fails to make a reasonable balance between the local and global search. In contrast, our original algorithm can succeed in solving these test functions consistently.

Furthermore, based on our observation, DyHF with dynamic implementation of the global and local search models converges

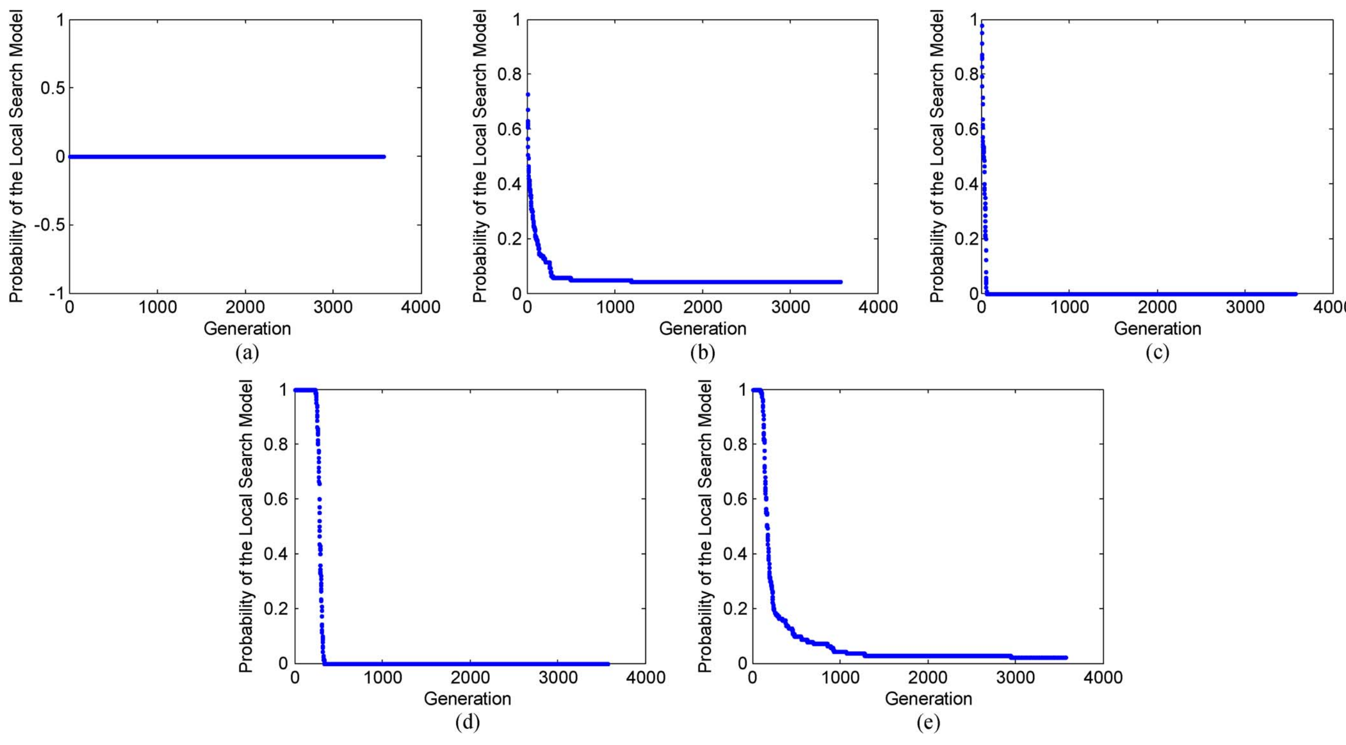

Fig. 4. Probability of the local search model versus generation for test functions g02, g04, g12, g14, and g15.

significantly faster than DyHF with constant implementation of the global and local search models, which signifies that the latter cannot properly distribute available computational resource during the evolution.

To further analyze the behavior and principle of DyHF, we record the probability of the local search model of a typical run in Fig. 4 for five test functions: g02, g04, g12, g14, and g15. For test function g02, the probability of the local search model is equal to zero during the evolution [Fig. 4(a)]. It is because g02 has a relatively large proportion of the feasible region (99.9971%), and all the individuals in the population are feasible during the evolution. Actually, g02 has a very rugged fitness landscape and is mainly used to investigate the global search ability of an algorithm. For test functions g04 and g12, the initial population consists of both the feasible and infeasible solutions [Fig. 4(b) and (c)]. As a result, the probability of the local search model is less than 1 at the beginning. Along with the evolution, the probability of the local search model decreases drastically due to the fact that more and more individuals in the population become feasible. Hereafter, the global search model plays a major role in searching for the global optimum in the feasible region. When the procedure terminates, the probability of the local search model is slightly bigger than zero and equal to zero for test functions g04 and g12, respectively. For test functions g14 and g15, since the initial population is entirely composed of infeasible solution, the probability of the local search model is equal to 1 at the beginning [Fig. 4(d) and (e)]. By making use of the local search model, the population is able to approach or enter the feasible region promptly. Therefore, the probability of the local search model is less than 1 after some generations. Afterward, the convergence behaviors of test functions g14 and g15 are similar to that of test functions g12 and g04, respectively. The reason why the probability of the local search model is slightly bigger than zero in the end for some test functions (such as g04 and g15) is that the newly generated offspring cannot Pareto dominate some infeasible solutions in the population, and, as a result, such infeasible solutions are kept in the population consistently. According to the above discussion, one can conclude that at different stages of evolution, different probabilities for the local and global search may be required to achieve the best performance.

B. Can DyHF Solve Unconstrained Single-Objective Optimization Problems?

Unlike COPs, in which the search space consists of both the feasible region and the infeasible region, for unconstrained single-objective optimization problems (USOPs) all the individuals in the search space are feasible. Therefore, when solving USOPs, the population contains feasible solutions only (i.e., $NF = NP$). Under this condition, DyHF only includes the global search model, and the local search model is not applied. It is noteworthy that $DE/rand/1/bin$, one of the classic versions of DE, is considered as the global search model in DyHF. Thus, DyHF can be used to solve USOPs directly, since in this case, the degree of constraint violation of an individual is equal to 0, and the comparison of individuals is based only on the objective function value [like equation (9)].

C. Effect of the Parameter Settings

1) Effect of NP: We study the influence of the population size NP on the performance using $NP = 80, 120, 140, 160,$ and 200. Table IX compares the experimental

TABLE IX
MEAN OBJECTIVE FUNCTION VALUE, SUCCESS RATE, AND FEASIBLE RATE ON TEST FUNCTIONS G02, G13, G17, G21, AND G23 WITH VARYING NP OVER 25 RUNS

Prob.	Mean objective function value (Success rate) [Feasible rate]				
	80	120	140	160	200
g02	-0.7998 (64%) [100%]	-0.8027 (92%) [100%]	-0.8036 (100%) [100%]	-0.8036 (100%) [100%]	-0.8036 (100%) [100%]
g13	0.0693 (96%) [100%]	0.0539 (100%) [100%]	0.0539 (100%) [100%]	0.0539 (100%) [100%]	0.0539 (100%) [100%]
g17	8856.4961 (96%) [100%]	8853.5339 (100%) [100%]	8853.5339 (100%) [100%]	8853.5339 (100%) [100%]	8853.8590 (92%) [100%]
g21	2.238468 (80%) [88%]	204.2027 (92%) [100%]	193.7245 (100%) [100%]	193.7245 (100%) [100%]	193.7245 (100%) [100%]
g23	-390.3170 (80%) [100%]	-376.0544 (92%) [100%]	-400.0551 (100%) [100%]	-400.0551 (100%) [100%]	-400.0551 (100%) [100%]

TABLE X
MEAN OBJECTIVE FUNCTION VALUE, SUCCESS RATE, AND FEASIBLE RATE ON TEST FUNCTIONS G03, G17, G21, AND G23 WITH VARYING NS OVER 25 RUNS

Prob.	Mean objective function value (Success rate) [Feasible rate]				
	6	8	10	12	14
g13	0.0693 (96%) [100%]	0.0539 (100%) [100%]	0.0539 (100%) [100%]	0.0539 (100%) [100%]	0.0539 (100%) [100%]
g17	8871.6495 (72%) [100%]	8853.9207 (92%) [100%]	8853.5339 (100%) [100%]	8853.5339 (100%) [100%]	8853.5339 (100%) [100%]
g21	193.7245 (100%) [100%]	193.7245 (100%) [100%]	193.7245 (100%) [100%]	193.7245 (100%) [100%]	204.2027 (92%) [100%]
g23	-400.0550 (100%) [100%]	-400.0551 (100%) [100%]	-400.0551 (100%) [100%]	-400.0551 (100%) [100%]	-376.0543 (92%) [100%]

TABLE XI
MEAN OBJECTIVE FUNCTION VALUE, SUCCESS RATE, AND FEASIBLE RATE ON TEST FUNCTIONS G01, G17, G21, AND G23 WITH VARYING F_1 OVER 25 RUNS

Prob.	Mean objective function value (Success rate) [Feasible rate]				
	0.5	0.6	0.7	0.8	0.9
g01	-14.6993 (88%) [100%]	-15.0000 (100%) [100%]	-15.0000 (100%) [100%]	-15.0000 (100%) [100%]	-15.0000 (100%) [100%]
g17	8857.4299 (72%) [100%]	8856.4985 (92%) [100%]	8853.5338 (100%) [100%]	8853.5338 (100%) [100%]	8853.5338 (100%) [100%]
g21	225.6249 (60%) [64%]	199.9219 (96%) [96%]	193.7245 (100%) [100%]	193.7245 (100%) [100%]	193.7245 (100%) [100%]
g23	-364.0540 (88%) [100%]	-400.0551 (96%) [100%]	-400.0551 (100%) [100%]	-400.0551 (100%) [100%]	-400.0551 (100%) [100%]

TABLE XII
MEAN OBJECTIVE FUNCTION VALUE, SUCCESS RATE, AND FEASIBLE RATE ON TEST FUNCTIONS G10, G14, G17, G21, AND G23 WITH VARYING C_{r1} OVER 25 RUNS

Prob.	Mean objective function value (Success rate) [Feasible rate]				
	0.6	0.7	0.8	0.9	1.0
g10	7049.3222 (4%) [100%]	7049. 3086 (4%) [100%]	7049.2660 (40%) [100%]	7049.2480 (100%) [100%]	7049.2480 (100%) [100%]
g14	-47.5691 (0%) [100%]	-47.6272 (0%) [8%]	-47.6392 (0%) [0%]	-47.7649 (100%) [100%]	-47.7649 (100%) [100%]
g17	8874.3415 (32%) [100%]	8870.2665 (36%) [100%]	8853.8862 (80%) [100%]	8853.6963 (88%) [100%]	8853.5339 (100%) [100%]
g21	251.2638 (52%) [100%]	225.1657 (76%) [100%]	198.9636 (96%) [100%]	198.9696 (100%) [100%]	193.7245 (100%) [100%]
g23	-239.2544 (0%) [32%]	-867.9987 (0%) [4%]	-101.3629 (0%) [0%]	-388.0547 (96%) [100%]	-400.0551 (100%) [100%]

results of test functions g02, g13, g17, g21, and g23. From Table IX, it seems that a smaller value for NP (i.e., 80) will lead to infeasible convergence, see, for example, test function g21. In addition, smaller values (i.e., 80 and 120) also result in premature convergence for some test functions. On the other hand, when the population size NP is relatively bigger (i.e., 200), incomplete convergence occurs for test function g17. The above results verify that NP can be chosen from 140 to 160.

2) Effect of NS: The effect of the subpopulation size NS on the performance is investigated by using different values: 6, 8, 10, 12, and 14. The experimental results of test functions g03, g17, g21, and g23 are summarized in Table X.

As shown in Table X, the algorithms cannot consistently reach the optimal solutions for test functions g13 and g17 and for test function g17, respectively, when $NS = 6$ and 8, which are slightly smaller subpopulation sizes. However, it is necessary to note that a relatively greater value for the subpopulation size (i.e., 14) also deteriorates the performance, since the algorithm has the difficulty in consistently finding the optimal solutions for test functions g21 and g23. When $NS =$ 10 and 12, the algorithms have similar overall performance, which suggests the range of [10, 12] for the subpopulation size.

3) Effect of F_1: We experiment with different scaling factor F_1: 0.5, 0.6, 0.7, 0.8, and 0.9. Results of test functions g01, g17, g21, and g23 are presented in Table XI.

For smaller values of F_1 (i.e., 0.5 and 0.6), the algorithms cannot find the feasible solutions in all runs for test function g21 and cannot converge in all trials for some test functions. It seems that no performance difference is observed when $F_1 = 0.7, 0.8,$ and 0.9. However, we find that the convergence speed will degrade with the increase of F_1.

4) Effect of C_{r1}: Table XII summarizes the results of test functions g10, g14, g17, g21, and g23 in the case of the crossover control parameter C_{r1} alone being changed to 0.6, 0.7, 0.8, 0.9, and 1.0.

TABLE XIII
MEAN OBJECTIVE FUNCTION VALUE, SUCCESS RATE, AND FEASIBLE RATE ON TEST FUNCTIONS G02, G03, G05, G07, G09, G10, G13, G14, G17, G19, G21, AND G23 WITH VARYING F_2 OVER 25 RUNS

Prob.	Mean objective function value (Success rate) [Feasible rate]				
	0.3	0.4	0.5	0.6	0.7
g02	-0.7795 (0%) [100%]	-0.7994 (64%) [100%]	-0.8036 (100%) [100%]	-0.8036 (100%) [100%]	-0.8036 (100%) [100%]
g03	-0.9998 (72%) [100%]	-1.0005 (100%) [100%]	-1.0005 (100%) [100%]	-1.0005 (100%) [100%]	-1.0005 (100%) [100%]
g05	5126.5093 (92%) [100%]	5126.4967 (100%) [100%]	5126.4967 (100%) [100%]	5126.4967 (100%) [100%]	5126.4967 (100%) [100%]
g07	24.3561 (0%) [100%]	24.3062 (100%) [100%]	24.3062 (100%) [100%]	24.3062 (100%) [100%]	24.3062 (100%) [100%]
g09	680.6342 (12%) [100%]	680.6301 (100%) [100%]	680.6301 (100%) [100%]	680.6301 (100%) [100%]	680.6301 (100%) [100%]
g10	7061.1862 (0%) [100%]	7049.2480 (100%) [100%]	7049.2480 (100%) [100%]	7049.2480 (100%) [100%]	7049.2480 (100%) [100%]
g13	0.0716 (64%) [100%]	0.0667 (92%) [100%]	0.0539 (100%) [100%]	0.0539 (100%) [100%]	0.0539 (100%) [100%]
g14	-47.7209 (0%) [100%]	-47.7649 (100%) [100%]	-47.7649 (100%) [100%]	-47.7649 (100%) [100%]	-47.7649 (100%) [100%]
g17	8864. 8240 (4%) [100%]	8861.9946 (4%) [100%]	8853.5339 (100%) [100%]	8853.5339 (100%) [100%]	8853.5339 (100%) [100%]
g19	44.1860 (0%) [100%]	32.6556 (100%) [100%]	32.6556 (100%) [100%]	32.6556 (100%) [100%]	32.6556 (100%) [100%]
g21	206.7320 (8%) [100%]	200.8411 (60%) [100%]	193.7245 (100%) [100%]	193.7245 (100%) [100%]	193.7245 (100%) [100%]
g23	-373.0115 (0%) [100%]	-371.8561 (12%) [100%]	-400.0551 (100%) [100%]	-400.0551 (100%) [100%]	-400.0551 (100%) [100%]

TABLE XIV
MEAN OBJECTIVE FUNCTION VALUE, SUCCESS RATE, AND FEASIBLE RATE ON TEST FUNCTIONS G01, G02, G05, G07, G09, G10, G14, G17, G19, G21, AND G23 WITH VARYING $P_{c_{r2}}$ OVER 25 RUNS

Prob.	Mean objective function value (Success rate) [Feasible rate]			
	0.0	0.5	0.75	1.0
g01	-15.0000 (100%) [100%]	-15.0000 (100%) [100%]	-15.0000 (100%) [100%]	-14.9992 (8%) [100%]
g02	-0.8022 (0%) [100%]	-0.8036 (100%) [100%]	-0.8036 (100%) [100%]	-0.7522 (0%) [100%]
g05	5126.4983 (88%) [100%]	5126.4967 (100%) [100%]	5126.4967 (100%) [100%]	5126.4967 (100%) [100%]
g07	24.3073 (0%) [100%]	24.3062 (100%) [100%]	24.3062 (100%) [100%]	24.3062 (100%) [100%]
g09	680.6317 (8%) [100%]	680.6301 (100%) [100%]	680.6301 (100%) [100%]	680.6301 (100%) [100%]
g10	7059.6450 (0%) [100%]	7049.2480 (100%) [100%]	7049.2480 (100%) [100%]	7049.2480 (100%) [100%]
g14	-47.7552 (0%) [100%]	-47.7649 (100%) [100%]	-47.7649 (100%) [100%]	-47.7649 (100%) [100%]
g17	8866.5793 (0%) [100%]	8853.8872 (80%) [100%]	8853.5339 (100%) [100%]	8853.5339 (100%) [100%]
g19	51.5006 (0%) [100%]	32.6556 (100%) [100%]	32.6556 (100%) [100%]	32.6556 (100%) [100%]
g21	198.7903 (8%) [100%]	193.7245 (100%) [100%]	193.7245 (100%) [100%]	225.1953 (76%) [100%]
g23	-379.9706 (0%) [100%]	-400.0551 (100%) [100%]	-400.0551 (100%) [100%]	-338.0538 (92%) [100%]

In the case of $C_{r1} = 0.6, 0.7,$ and 0.8, the performance of the algorithms is irregular as shown in Table XII, since the success rates gradually increase with the increase of C_{r1} for test functions g10, g17, and g21, but the feasible rates gradually decrease with the increase of C_{r1} for test function g14 and g23. In the case of $C_{r1} = 0.9$, the overall performance of the algorithm significantly outperforms that of $C_{r1} = 0.6, 0.7,$ and 0.8. However, the algorithm still fails to solve test functions g17 and g23 consistently when $C_{r1} = 0.9$. In the case of $C_{r1} = 1.0$, the algorithm exhibits the best overall performance. We believe that it is because the diversity of the population is very good when $C_{r1} = 1.0$ since the offspring is totally inherited from the mutant vector. Hence, we conclude that significant improvements are obtained when $C_{r1} = 1.0$.

5) Effect of F_2: The effect of the scaling factor F_2 in the global search model has been demonstrated by carrying out different experiments in which all the test functions are studied at $F_2 = 0.3, 0.4, 0.5, 0.6,$ and 0.7. The results of test functions g02, g03, g05, g07, g09, g10, g13, g14, g17, g19, g21, and g23 are summarized in Table XIII.

From Table XIII, it can be seen that the overall performance gradually becomes better with the scaling factor F_2 being increased from 0.3 to 0.5. It seems that the algorithms show similar overall performance when $F_2 = 0.5, 0.6,$ and 0.7. However, based on our observation, the increase of F_2 has a negative influence on the convergence speed.

6) Effect of $P_{c_{r2}}$: Finally, Table XIV summarizes the experimental results of test functions g01, g02, g05, g07, g09, g10, g14, g17, g19, g21, and g23 with different $P_{c_{r2}}$: 0.0, 0.5, 0.75, and 1.0. Note that $P_{c_{r2}} = 0.0$ and $P_{c_{r2}} = 1.0$ mean $C_{r2} = 0.1$ and $C_{r2} = 1.0$ during the evolution, respectively.

In the case of $P_{c_{r2}} = 0.0$, the algorithm cannot consistently converge to the optimal solutions for all the selected test functions except for test function g01. Also, in the case of $P_{c_{r2}} = 1.0$, the algorithm cannot consistently find the optimal solutions for test functions g01, g02, g21, and g23. This is because when $P_{c_{r2}} = 0.0$, the algorithm focuses on exploitation, and the global search ability becomes very poor, and when $P_{c_{r2}} = 1.0$, the algorithm focuses on exploration, and the convergence speed is not good. Thus, the overall performance of the algorithms is not good when $P_{c_{r2}} = 0.0$ and 1.0. In the case of $P_{c_{r2}} = 0.5$, the overall performance of the algorithm is significantly better than the above two cases. This can be attributed to the fact that under this condition, the algorithm can balance the exploration and exploitation to a certain degree. However, the performance of the algorithm with $P_{c_{r2}} = 0.5$ can be further improved by increasing $P_{c_{r2}}$ to 0.75, especially for test function g17. Therefore, $P_{c_{r2}} = 0.75$ is a good choice for DyHF.

VII. Conclusion

To overcome the drawbacks and improve the performance of HOCEA [4], a *dynamic hybrid framework*, referred as DyHF, has been proposed in this paper to solve COPs. In DyHF, the global search model is combined with the local search model to search the optimal solution. During the evolution, DE is considered as the search engine in these two kinds of models. A remarkable difference between our method and other existing methods is that in our method, the global and local search models are executed dynamically based on the feasibility proportion of the current population.

The performance of DyHF has been tested on 22 benchmark test functions. Based on the experimental results, DyHF has the capability to solve all the test functions successfully. The overall performance of DyHF is very competitive with six state-of-the-art approaches in the community of constrained evolutionary optimization. In addition, the effectiveness of dynamic implementation of the global and local search models and the effect of the parameters on the performance have been studied by experiments.

The source code of DyHF is written in MATLAB and can be obtained from the first author upon request.

Acknowledgment

The authors sincerely thank the anonymous associate editor and the anonymous reviewers for their very helpful and constructive comments and suggestions.

References

[1] Z. Michalewicz and M. Schoenauer, "Evolutionary algorithm for constrained parameter optimization problems," *Evol. Comput.*, vol. 4, no. 1, pp. 1–32, Feb 1996.

[2] C. A. Coello Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art," *Comput. Methods Appl. Mech. Eng.*, vol. 191, no. 11/12, pp. 1245–1287, Jan. 2002.

[3] Z. Cai and Y. Wang, "A multiobjective optimization-based evolutionary algorithm for constrained optimization," *IEEE Trans. Evol. Comput.*, vol. 10, no. 6, pp. 658–675, Dec. 2006.

[4] Y. Wang, Z. Cai, G. Guo, and Y. Zhou, "Multiobjective optimization and hybrid evolutionary algorithm to solve constrained optimization problems," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 37, no. 3, pp. 560–575, Jun. 2007.

[5] S. Tsutsui, M. Yamamure, and T. Higuchi, "Multi-parent recombination with simplex crossover in real coded genetic algorithms," in *Proc. Genetic Evol. Comput. Conf.*, 1999, pp. 657–664.

[6] R. Storn and K. Price, "Differential evolution—A simple and efficient adaptive scheme for global optimization over continuous spaces," Int. Comput. Sci. Inst., Berkeley, Tech. Rep. TR-95-012, 1995.

[7] J. J. Liang, T. P. Runarsson, E. Mezura-Montes, M. Clerc, P. N. Suganthan, C. A. Coello Coello, and K. Deb, "Problem definitions and evaluation criteria for the CEC 2006," Nanyang Technol. Univ., Singapore, Tech. Rep., 2006.

[8] T. Takahama and S. Sakai, "Constrained optimization by the ε constrained differential evolution with gradient-based mutation and feasible elites," in *Proc. Congr. Evol. Comput.*, Vancouver, BC, Canada, 2006, pp. 1–8.

[9] T. Takahama and S. Sakai, "Constrained optimization by applying the α constrained method to the nonlinear simplex method with mutations," *IEEE Trans. Evol. Comput.*, vol. 9, no. 5, pp. 437–451, Oct. 2005.

[10] J. J. Liang and P. N. Suganthan, "Dynamic multi-swarm particle swarm optimizer with a novel constraint-handling mechanism," in *Proc. Congr. Evol. Comput.*, Vancouver, BC, Canada, 2006, pp. 9–16.

[11] J. Brest, V. Zumer, and M. S. Maucec, "Self-adaptive differential evolution algorithm in constrained real-parameter optimization," in *Proc. Congr. Evol. Comput.*, Vancouver, BC, Canada, 2006, pp. 215–222.

[12] J. Brest, B. Boskovic, S. Greiner, V. Zumer, and M. Sepesy Maucec, "Performance comparison of self-adaptive and adaptive differential evolution algorithms," Univ. Maribor, Faculty Elect. Eng. Comput. Sci., Maribor, Slovenia, Tech. Rep. 2006-1-LABRAJ, 2006.

[13] K. Zielinski and R. Laur, "Constrained single-objective optimization using differential evolution," in *Proc. Congr. Evol. Comput.*, Vancouver, BC, Canada, 2006, pp. 223–230.

[14] K. Zielinski and R. Laur, "Constrained single-objective optimization using particle swarm optimization," in *Proc. Congr. Evol. Comput.*, Vancouver, BC, Canada, 2006, pp. 443–450.

[15] M. F. Tasgetiren and P. N. Suganthan, "A multi-populated differential evolution algorithm for solving constrained optimization problem," in *Proc. Congr. Evol. Comput.*, Vancouver, BC, Canada, 2006, pp. 33–40.

[16] A. E. Munoz-Zavala, A. Hernandez-Aguirre, E. R. Villa-Diharce, and S. Botello-Rionda, "PESO+ for constrained optimization," in *Proc. Congr. Evol. Comput.*, Vancouver, BC, Canada, 2006, pp. 231–238.

[17] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. Coello Coello, "Modified differential evolution for constrained optimization," in *Proc. Congr. Evol. Comput.*, Vancouver, BC, Canada, 2006, pp. 25–32.

[18] V. L. Huang, A. K. Qin, and P. N. Suganthan, "Self-adaptive differential evolution algorithm for constrained real-parameter optimization," in *Proc. Congr. Evol. Comput.*, Vancouver, BC, Canada, 2006, pp. 17–24.

[19] T. P. Runarsson, "Approximate evolution strategy using stochastic ranking," in *Proc. Congr. Evol. Comput.*, Vancouver, BC, Canada, 2006, pp. 745–752.

[20] T. P. Runarsson and X. Yao, "Stochastic ranking for constrained evolutionary optimization," *IEEE Trans. Evol. Comput.*, vol. 4, no. 3, pp. 284–294, Sep. 2000.

[21] S. Kukkonen and J. Lampinen, "Constrained real-parameter optimization with generalized differential evolution," in *Proc. Congr. Evol. Comput.*, Vancouver, BC, Canada, 2006, pp. 207–214.

[22] A. Sinha, A. Srinivasan, and K. Deb, "A population-based, parent centric procedure for constrained real-parameter optimization," in *Proc. Congr. Evol. Comput.*, Vancouver, BC, Canada, 2006, pp. 239–245.

[23] L. C. Cagnina, S. C. Esquivel, and C. A. Coello Coello, "A bi-population PSO with a shake-mechanism for solving constrained numerical optimization," in *Proc. Congr. Evol. Comput.*, Singapore, Sep. 2007, pp. 670–676.

[24] G. Leguizamón and C. A. Coello Coello, "A boundary search based ACO algorithm coupled with stochastic ranking," in *Proc. Congr. Evol. Comput.*, Singapore, Sep. 2007, pp. 165–172.

[25] E. Mezura-Montes and B. Cecilia-López-Ramírez, "Comparing bio-inspired algorithms in constrained optimization problems," in *Proc. Congr. Evol. Comput.*, Singapore, 2007, pp. 662–669.

[26] F. Huang, L. Wang, and Q. He, "An effective co-evolutionary differential evolution for constrained optimization," *Appl. Math. Comput.*, vol. 186, no. 1, pp. 340–356, Mar 2007.

[27] R. Mallipeddi and P. N. Suganthan, "Evaluation of novel adaptive evolutionary programming on four constraint handling techniques," in *Proc. Congr. Evol. Comput.*, Hong Kong, Jun. 2008, pp. 4045–4052.

[28] Y. Wang, Z. Cai, Y. Zhou, and W. Zeng, "An adaptive trade-off model for constrained evolutionary optimization," *IEEE Trans. Evol. Comput.*, vol. 12, no. 1, pp. 80–92, Feb. 2008.

[29] A. S. S. , B. Ullah, R. Sarker, D. Cornforth, and C. Lokan, "AMA: A new approach for solving constrained real-valued optimization problems," *Soft Comput.*, vol. 13, no. 8/9, pp. 741–762, Mar. 2009.

[30] Z. Huang, C. Wang, and H. Tian, "A genetic algorithm with constrained sorting method for constrained optimization problems," in *Proc. IEEE Int. Conf. Intell. Comput. Intell. Syst.*, 2009, pp. 806–811.

[31] L. Li, L. Wang, and Y. Xu, "Differential evolution with level comparison for constrained optimization," in *Proc. ICIC*, vol. 5755/2009, *Lecture Notes Computer Science*, 2009, pp. 351–360.

[32] B. Tessema and G. G. Yen, "An adaptive penalty formulation for constrained evolutionary optimization," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 39, no. 3, pp. 565–578, May 2009.

[33] L. Jia, G. Zou, C. Luo, and J. Zou, "BIEA: A novel evolutionary algorithm for nonlinear constrained programming," in *Proc. 2nd Int. Asia Conf. Informatics Control, Autom. Robot.*, 2010, pp. 87–90.

[34] T. Ray and K. M. Liew, "Society and civilization: An optimization algorithm based on the simulation of social behavior," *IEEE Trans. Evol. Comput.*, vol. 7, no. 4, pp. 386–396, Aug. 2003.

Yong Wang (M'08) was born in Hubei, China, in 1980. He received the B.S. degree in automation from the Wuhan Institute of Technology, Wuhan, China, in 2003, and the M.S. degree in pattern recognition and intelligent systems and the Ph.D. degree in control science and engineering from the Central South University (CSU), Changsha, China, in 2006 and 2011, respectively.

Currently, he is a Lecturer with the School of Information Science and Engineering, CSU. His main research interests include evolutional computation, differential evolution, global optimization, constrained optimization, multiobjective optimization, and their real-world applications. He has published several papers in the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, *Evolutionary Computation* (MIT Press), and the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, PART B: CYBERNETICS. He has been a reviewer for journals such as IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, PART B: CYBERNETICS.

Dr. Wang is a member of the IEEE Task Force on Nature-Inspired Constrained Optimization.

Zixing Cai (SM'98) received the Diploma degree from the Department of Electrical Engineering, Jiao Tong University, Xi'an, China, in 1962.

He has been teaching and doing research at the School of Information Science and Engineering, Central South University, Changsha, China, since 1962. From May 1983 to December of 1983, he visited the Center of Robotics, Department of Electrical Engineering and Computer Science, University of Nevada, Reno. He visited the Advanced Automation Research Laboratory, School of Electrical Engineering, Purdue University, West Lafayette, IN, from December 1983 to June 1985. From October 1988 to September 1990, he was a Senior Research Scientist with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Science, Beijing, China, and the National Laboratory of Machine Perception, Center of Information, Beijing University, Beijing, China. Since February 1989, he has become an Expert of United Nations granted by United Nations Industrial Development Organization. From September 1992 to March 1993, he visited the NASA Center for Intelligent Robotic Systems for Space Exploration, and the Department of Electrical, Computer and System Engineering, Rensselaer Polytechnic Institute, Troy, NY, as a Visiting Professor. From April 2004 to July 2004, he visited the Institute of Informatics and Automation, Russia Academy of Sciences, Moscow, Russia. From April 2007 to August 2007, he visited Denmark Technical University, Lyngby, Denmark, as a Visiting Professor. He has authored/coauthored over 600 papers and 30 books/textbooks. His current research interests include intelligent systems, artificial intelligence, intelligent computation, and robotics.

Prof. Cai received over 30 state, province, university awards in science, technology, and teaching. One of the most recent prizes is the State Eminent Professor Prize of China.