6-1-2018

Shuffled Complex-Self Adaptive Hybrid EvoLution (SC-SAHEL) Optimization Framework

Mojtaba Sadegh
Boise State University

Publication Information

Sadegh, Mojtaba. (2018). "Shuffled Complex-Self Adaptive Hybrid EvoLution (SC-SAHEL) Optimization Framework". *Environmental Modelling & Software, 104*, 215-235. http://dx.doi.org/10.1016/j.envsoft.2018.03.019

Shuffled Complex-Self Adaptive Hybrid EvoLution (SC-SAHEL) Optimization Framework

Matin Rahnamay Naeini[1], Tiantian Yang[1,2], Mojtaba Sadegh[1,3], Amir Aghakouchak[1], Kuo-lin Hsu[1] , Soroosh Sorooshian[1], Qingyun Duan[4], and Xiaohui Lei[5]

[1]Center for Hydrometeorology and Remote Sensing (CHRS) & Department of Civil and Environmental Engineering, University of California, Irvine, California, USA.

[2]Deltares USA Inc., Silver Spring, Maryland, USA

[3]Department of Civil Engineering, Boise State University, Boise, Idaho, USA.

[4]Beijing Normal University, Faculty of Geographical Sciences. Beijing, China

[5]China Institute of Water Resources and Hydropower Research, Beijing, China

Corresponding author: Tiantian Yang (tiantiay@uci.edu)

Abstract

Simplicity and flexibility of Meta-Heuristic optimization algorithms have attracted lots of attention in the field of optimization. Different optimization methods, however, hold algorithm-specific strengths and limitations, and selecting best-performing algorithm for a specific problem is a tedious task. We introduce a new hybrid optimization framework, entitled Shuffled Complex-Self Adaptive Hybrid EvoLution (SC-SAHEL), which combines strengths of different Evolutionary Algorithms (EAs) in a parallel computing scheme. SC-SAHEL explores performance of different EAs, i.e., capability to escape local attractions, speed, convergence, etc., during population evolution as each individual EA suits differently to various response surfaces. The SC-SAHEL algorithm is benchmarked over 29 conceptual test functions, and a real-world case - hydropower reservoir model. Results show that the SC-SAHEL algorithm is rigorous and effective in finding global optimum for a majority of test cases, and computationally efficient comparing to individual EAs.

Keywords

Shuffled Complex Evolution (SCE); Hybrid Optimization; Evolutionary Algorithm (EA); Reservoir Operation; Hydropower

Software availability

Name of software: SC-SAHEL

Developer: Matin Rahnamay Naeini

Contact address: rahnamam@uci.edu

Program language: MATLAB

Year first available: 2018

Availability: Freely available to public at chrs.web.uci.edu/software.php and MathWorks website

Software requirements: MATLAB 9.0

1 Introduction

Meta-Heuristic optimization algorithms have gained a great deal of attention in science and engineering (Blum and Roli 2003, Boussaïd et al. 2013, Lee and Geem 2005, Maier et al. 2014, Nicklow et al. 2010, Reed et al. 2013). Simplicity and flexibility of these algorithms, along with their robustness make them attractive tools for solving optimization problems (Coello et al. 2007, Lee and Geem 2005). Many of the meta-heuristic algorithms are inspired by a physical phenomenon, such as animals social and foraging behavior and natural selection. For example, Simulated Annealing (Kirkpatrick et al. 1983), Big Bang-Big Crunch (Erol and Eksin 2006), Gravitational Search Algorithm (Rashedi et al. 2009), Charged System Search (Kaveh and Talatahari 2010) are inspired by various physical phenomena. Ant Colony Optimization (Dorigo et al. 1996), Particle Swarm Optimization (Kennedy 2010), Bat-inspired Algorithm (Yang 2010), Firefly Algorithm (Yang 2009), Dolphin Echolocation (Kaveh and Farhoudi 2013), Grey Wolf Optimizer (Mirjalili et al. 2014), Bacterial Foraging (Passino 2002), Genetic Algorithm (Golberg 1989, Holland 1992), and Differential Evolution (Storn and Price 1997) are examples of algorithms inspired by animal's social and foraging behavior, and the natural selection mechanism of Darwin's Evolution Theorem. According to the No-Free-Lunch (NFL) (Wolpert and Macready 1997) theorem, none of these algorithms are consistently superior to others over a variety of problems, although some of them may outperform on a certain type of optimization problem.

The NFL theorem has been a source of motivation for developing hybrid optimization algorithms (Mirjalili et al. 2014, Woodruff et al. 2013). It has encouraged scientists and researchers to combine the strengths of different algorithms and devise more robust and efficient optimization algorithms that suit a broad class of problems (Qin and Suganthan 2005, Vrugt and Robinson 2007, Vrugt et al. 2009, Hadka and Reed 2013, Sadegh et al. 2017). These efforts led to emergence of multi-method and self-adaptive optimization algorithms such as Self-adaptive DE algorithm

(SaDE) (Qin and Suganthan 2005), A Multialgorithm Genetically Adaptive Method for Single Objective Optimization (AMALGAM-SO) (Vrugt and Robinson 2007, Vrugt et al. 2009) and Borg (Hadka and Reed 2013). They all reguarly update the search mechanism during the course of optimization according to the information obtained from the response surface.

Here, we propose a new self-adaptive hybrid optimization framework, entitled Shuffled Complex-Self Adaptive Hybrid EvoLution (SC-SAHEL). The SC-SAHEL framework employs multiple Evolutionary Algorithms (EAs) as search cores, and enables competition among different algorithms as optimization run progresses. The proposed framework differs from other multi-method algorithms as it grants independent evolution of population by each EA. In this framework, population is partitioned into equally sized groups, so-called complexes; each assigned to different EAs. Number of complexes assigned to each EA is regularly updated according to their performance. In general, the newly developed framework has two main characteristics. First, all the EAs evolve population in a parallel structure. Second, each participating EA works independent of other EAs. The architecture of SC-SAHEL is inspired by the concept of the Shuffled Complex Evolution algorithm - University of Arizona (SCE-UA) (Duan et al. 1992). The SCE-UA algorithm is a population-evolution based algorithm (Madsen 2003), which evolves individuals by partitioning population into different complexes. The complexes are evolved for a specific number of iterations independent of other complexes, and then are forced to shuffle.

The SCE-UA framework employs Nelder-Mead simplex (Nelder and Mead 1965) technique along with the concept of controlled random search (Price 1987), clustering (Kan and Timmer 1987), competitive evolution (Holland 1975) and complex shuffling (Duan et al. 1993) to offer a global optimization strategy. By employing these techniques, the SCE-UA algorithm provides a robust optimization framework and has shown numerically to be competitive and efficient comparing to other algorithms, such as GA, for calibrating rainfall-runoff models (Beven

2011, Gan and Biftu 1996, Wagener et al. 2004, Wang et al. 2010). The SCE-UA algorithm has been widely used in water resources management (Barati et al. 2014, Eckhardt and Arnold 2001, K. Ajami et al. 2004, Lin et al. 2006, Liong and Atiquzzaman 2004, Madsen 2000, Sorooshian et al. 1993, Toth et al. 2000, Yang et al. 2015, Yapo et al. 1996), as well as other fields of study, such as pyrolysis modeling (Ding et al. 2016, Hasalová et al. 2016) and Artificial Intelligence (Yang et al. 2017).

Application of the SCE-UA is not limited to solving single objective optimization problems. The Multi-Objective Complex evolution, University of Arizona (MOCOM-UA), is an extension of the SCE-UA for solving multi-objective problems (Boyle et al. 2000, Yapo et al. 1998). Besides, the SCE-UA architecture has been used to develop Markov Chain Monte Carlo (MCMC) sampling, named Shuffled Complex Evolution Metropolis algorithm (SCEM-UA) and the Multi-Objective Shuffled Complex Evolution Metropolis (MOSCEM) to infer posterior parameter distribution of hydrologic models (Vrugt et al. 2003a, Vrugt et al. 2003b). The Metropolis scheme is used as the search kernel in the SCEM-UA and MOSCEM-UA (Chu et al. 2010, Vrugt et al. 2003a, Vrugt et al. 2003b). There is also an enhanced version of SCE-UA, which is developed by Chu et al. (2011) entitled the Shuffled Complex strategy with Principle Component Analysis, developed at the University of California, Irvine (SP-UCI). Chu et al. (2011) found that the SCE-UA algorithm may not converge to the best solution on high-dimensional problems due to “population degeneration” phenomenon. The “population degeneration” refers to the situation when the search particles span a lower dimension space than the original search space (Chu et al. 2010), which causes the search algorithm to fail in finding the global optimum. To address this issue, the SP-UCI algorithm employs Principle Component Analysis (PCA) in order to find and restore the missing dimensions during the course of search (Chu et al. 2011).

Both SCE-UA and SP-UCI start the evolution process by generating a population within the feasible parameters space. Then, population is partitioned into different complexes, and each complex is evolved independently. Each member of the complex has the potential to contribute to offspring in the evolution process. In each evolution step, more than two parents may contribute to generating offspring. To make the evolution process competitive, a triangular probability function is used to select parents. As a result, the fittest individuals will have a higher chance of being selected. Each complex is evolved for a specific number of iterations, and then complexes are shuffled to globally share the information attained by individuals during the search.

The Competitive Complex Evolution (CCE) and Modified Competitive Complex Evolution (MCCE) are the search cores of the SCE-UA and SP-UCI algorithm, respectively. The CCE and MCCE evolutionary processes are developed based on Nelder-Mead (Nelder and Mead 1965) method with some modification. The evolution process in the SCE-UA is not limited to these algorithms. In fact, several studies have incorporated different EAs into the structure of the SCE-UA algorithm. For example, the Frog Leaping (FL) is developed by adapting Particle Swarm Optimization (PSO) algorithm to the SCE-UA structure for solving discrete problems (Eusuff et al. 2006, Eusuff and Lansey 2003). Mariani et al. (2011) proposed an SCE-UA algorithm which employs DE for evolving the complexes. These studies revealed the flexibility of the SCE-UA in combination with other types of EAs; however, the potential of combining different algorithms into a hybrid shuffled complex scheme has not been investigated.

The unique structure of the SCE-UA algorithm along with the flexibility of the algorithm for using different EAs, motivated us to use the SCE-UA as the cornerstone of the SC-SAHEL framework. The SC-SAHEL algorithm employs multiple EAs for evolving the population in a similar structure as that of the SCE-UA, with the goal of selecting the most suitable search algorithm at each optimization step. On the one hand, some EAs are more capable of visiting the new regions of the search space and exploring the problem space, and hence are particularly suitable at the beginning of the optimization (Olorunda and Engelbrecht 2008). On the other hand, some EAs are more capable of searching within the visited regions of the search space, and hence boosting the convergence process after finding the region of interest (Mirjalili and Hashim 2010). Balancing between these two steps, which are referred to as exploration and exploitation (Moeini and Afshar 2009), is a challenging task in stochastic optimization methods (Črepinšek et al. 2013). The SC-SAHEL algorithm maintains a balance between exploration and exploitation phases by evaluating the performance of participating EAs at each optimization step. EAs contribute to the population evolution according to their performance in previous steps. The algorithms' performance is evaluated by comparing the evolved complexes before and after evolution. In this process, the most suitable algorithm for the problem space become the dominant search core.

In this study, four different EAs are used as search cores in the proposed SC-SAHEL framework, including Modified Competitive Complex Evolution (MCCE) used in the SP-UCI algorithm, Modified Frog Leaping (MFL), Modified Grey Wolf Optimizer (MGWO), and Differential Evolution (DE). To better illustrate the performance of the hybrid SC-SAHEL algorithm, the framework is benchmarked over 29 test functions and compared to SC-SAHEL with single EA. Among the 29 employed test functions, there are 23 classic test functions (Xin et al. 1999) and 6 composite test functions (Liang et al. 2005), which are commonly used as benchmarks in comparing optimization algorithms.

Furthermore, the SC-SAHEL framework is tested for a conceptual hydropower model, which is built for the Folsom reservoir located in the northern California, USA. The objective is to maximize the hydropower generation, by finding the optimum discharge from the reservoir. The study period covers run-off season in California from April to June, in which reservoirs have the highest annual storage volume (Field and Lund 2006). Using the proposed framework, we compared different EAs’ capability of finding a near-optimum solution for dry, wet, and below-normal scenarios. The results support that the proposed algorithm is not only competitive in terms of increasing power generation, but also is able to reveal the advantages and disadvantages of participating EAs.

The rest of the paper is organized as follow. In section 2, structure of the SC-SAHEL algorithm and details of four EAs are presented. Section 3 presents the test functions, settings of the experiments, and results obtained for each test function. Section 4 introduces the reservoir model and the optimization results for the case study. Finally, in section 5, we draw conclusion, summarize some limitations about the newly introduced framework, and suggest some directions for future work.

2 Methodology

The SC-SAHEL algorithm is a parallel optimization framework, which is built based on the original SCE-UA architecture. SC-SAHEL, however, differs from the original SCE-UA algorithm by using multiple search mechanisms instead of only employing the Nelder-Mead simplex downhill method. In this section, we first introduce the main structure of SC-SAHEL. Then, we present four different EAs, which are employed as search cores in the SC-SAHEL framework. These algorithms are selected for illustrative purpose only and can be replaced by other evolutionary algorithms. Some modifications are made to the original form of these algorithms, to allow fair competition between EAs. These modifications are detailed in the appendix A-D.

2.1 The SC-SAHEL framework

The proposed SC-SAHEL optimization strategy starts with generating a population with a pre-defined sampling method within feasible parameters' range. The framework supports user-defined sampling methods, besides built-in Uniform Random Sampling (URS) and Latin Hypercube Sampling (LHS). The population is then partitioned into different complexes. The partitioning process warrants maintaining diversity of population in each complex. In doing so, population is first sorted according to (objective) function values. Then, sorted population is divided into NGS equally-sized groups (NGS being the number of complexes), ensuring that members of each group have similar objective function values. Each complex subsequently will randomly select a member from each of these groups. This procedure maintains diversity of the population within each complex. The complexes are then assigned to EAs and evolved. In contrast to the original concept of the SCE-UA, the complexes are evolved with different EAs rather than single search mechanism. At the beginning of the search, an equal number of complexes is assigned to each evolutionary method. For instance, if population is partitioned into 8 complexes and 4 different EAs are used, each algorithm will evolve 2 complexes independently (2-2-2-2). After evolving the complexes for pre-specified number of steps, the Evolutionary Method Performance (EMP) metric (Eq.1) will be calculated for each EA,

$$\mathrm{EMP} = \frac{\mathrm{mean}(F) - \mathrm{mean}(F_N)}{\mathrm{mean}(F)}, \tag{1}$$

in which, F and F_N are objective function values of individuals in each complex before and after evolution, respectively.

The EMP metric measures change in the mean objective function value of individuals in each complex in comparison to their previous state. A higher EMP value indicates a larger reduction in the mean objective function value obtained by the individuals in the complex. The performance of each evolutionary algorithm is then evaluated based on the mean value of EMP calculated for each evolved complex. The algorithms are then ranked according to the EMP values. Ranks are in turn used to assign number of complexes to each evolutionary method for the next iteration. The highest ranked algorithm will be assigned an additional complex to evolve in the next shuffling step, while, the lowest ranked evolutionary algorithm will lose one complex for the next step. For instance, if all the EAs have 2 complexes to evolve (2-2-2-2 case), the number of complexes assigned to each EA can be updated to 3-2-2-1. In other words, this logic is an “award and punishment” process, in which the algorithm with best performances will be “awarded” with an additional complex to evolve in the next iteration, while the worst-performing algorithm will be “punished” by losing one complex.

It is worth mentioning that as some of the algorithms may have poor performance in the exploration phase, they might lose all their complexes during the adaptation process. This might be troublesome as these algorithms may be superior in the exploitation phase. If use of such algorithms are terminated in the exploration phase, they cannot be selected during the convergence steps. Hence, EAs termination is avoided to fully utilize the potential of EAs in all the optimization steps and balance the exploration and exploitation phases. The minimum number of complexes assigned to each evolutionary method is restricted to at least 1 complex in this case. If the lowest ranked EA has only 1 complex to evolve, it won’t lose its last complex. If an algorithm outperforms others throughout the evolution of complexes, the number of complexes assigned to the superior EA will be equal to the total number of complexes minus the number of EAs plus one. In this case, all other algorithms are evolving one complex only. As all algorithms are evolving at least one complex, they have the chance to outperform other EAs and gain more complexes during the optimization process, and to potentially become the dominant search method as the search continues toward exploitation phase. Figure 1 briefly shows the flowchart of the SC-SAHEL algorithm, pseudo code of which is as follows:

Step 0. Initialization. Select NGS > 1 and NPS (suggested NPS > 2n+1, where n is the dimension of the problem), where NGS is the number of complexes and NPS is the number of individuals in the complexes. NGS should be proportional to the number of evolutionary algorithms so that all the participating EAs have an equal number of complexes at the beginning of the search.

Step 1. Sample NPT points in the feasible parameter space using a user-defined sampling method, where NPT equals to NGS×NPS. Compute objective function value for each point.

Step 2. Rank and sort all individuals in the order of increasing objective function value.

Step 3. Partition the entire population into complexes. Assign complexes to the participating EAs.

Step 4. Monitor and restore population dimensionality using PCA algorithm (Optional).

Step 5. Evolve each complex using the corresponding EA.

Step 6. After evolving the complexes for a pre-defined number of iterations, calculate the mean EMP for each EA.

Step 7. Rank the participating EAs according to the mean EMP value of each evolutionary method. The highest ranked method will get additional complex in the next iteration, while the worst evolutionary method will lose one.

Step 8. Shuffle complexes and form a new population.

Step 9. Check whether the convergence criteria are satisfied, otherwise go to step 3.

SC-SAHEL allows for different settings that can influence the performance of the algorithm. Careful consideration should be devoted to the selection of these settings, including number of complexes, number of individuals within each complex, number of evolution steps before each shuffling, and stopping criteria thresholds. Some of these settings are adopted from the suggested settings for the SCE-UA. For instance, the number of points within each complex is set to $2d + 1$, where d is dimension of the problem. However, some of the suggested settings cannot be applied to the SC-SAHEL framework due to use of different EAs. These settings can be changed according to the complexity of the problem and the EAs used within the framework. For instance, the number of complexes, the number of points within each complex, and the number of evolution steps before each shuffling are problem dependent.

The SC-SAHEL framework employs three different stopping criteria which are adopted from SCE-UA and SP-UCI. These stopping criteria include number of function evaluations, range of samples that span the search space, and improvement in the objective function value in the last *m* shuffling steps. These criteria are compared to pre-defined thresholds, which can in turn be tuned according to the complexity of the problem. Improper selection of these thresholds may lead to early or delayed convergence.

2.2 Evolutionary algorithms used in the SC-SAHEL

In this paper, we employ four different EAs to illustrate the flexibility of the SC-SAHEL framework in adopting various EAs and show the algorithms competition. These algorithms are briefly presented here. The pseudo code and details of these algorithms can be found in Appendix A-D.

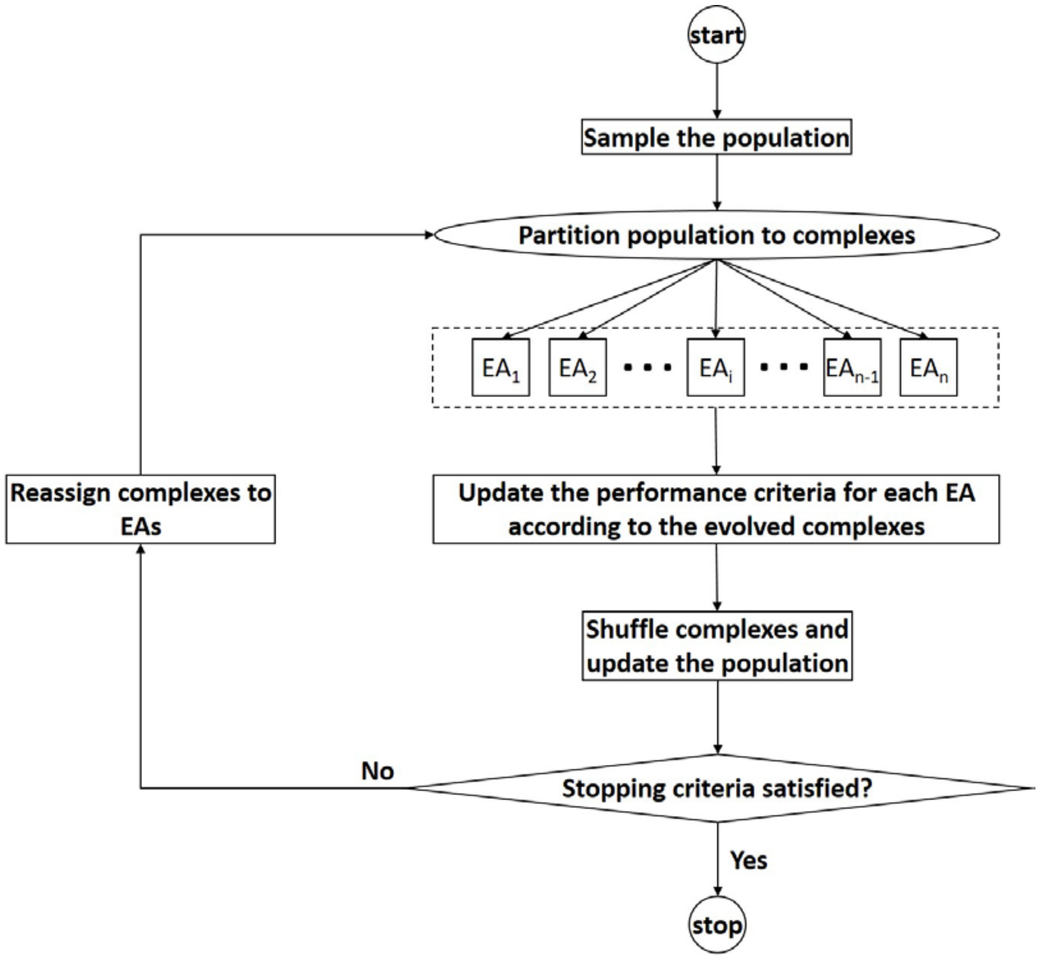

Figure 1. The SC-SAHEL framework flowchart

2.2.1 Modified Competitive Complex Evolution (MCCE)

The MCCE algorithm is an enhanced version of CCE algorithm used in the SCE-UA framework; which provides a robust, efficient, and effective EA for exploring and exploiting the search space. The MCCE algorithm is developed based on the Nelder-Mead algorithm, however, Chu et al. (2011) found that the shrink concept in the Nelder-Mead algorithm can cause premature convergence to a local optimum. Interested readers can refer to (Chu et al. 2010, 2011) for further details on MCCE algorithm. The pseudo code of the MCCE algorithm is detailed in Appendix A. SC-SAHEL has similar performance to SP-UCI, when the MCCE algorithm is used as the only search mechanism and PCA and resampling settings of SP-UCI are enabled. For simplification and comparison, SC-SAHEL with the MCCE algorithm as search core is referred as SP-UCI, hereafter.

2.2.2 Modified Frog Leaping (MFL)

The Frog Leaping (FL) algorithm uses adapted PSO algorithm as a local search tool within the SCE-UA framework (Eusuff and Lansey 2003). FL has shown to be an efficient search algorithm for discrete optimization problems, and can find optimum solution much faster as compared to the GA algorithm (Eusuff et al. 2006). In order to adapt the FL algorithm to the SC-SAHEL parallel framework, we introduce a slightly modified version of FL algorithm entitled MFL. Further details and pseudo code of the MFL can be found in Appendix B. The original FL algorithm and the MFL have four main differences. First, the original FL is designed for discrete optimization problems, however, the MFL is modified for continuous domain. Second, the modified FL uses the best point in the subcomplex for generating new points, however, in the original FL framework new points are generated using the best point in the complex and the entire population. The reason for this modification is to avoid using any external information by participating EAs. In other words, the amount of information given to each EAs is limited to the complex assigned to the EAs. Third, as the MFL algorithm only uses the best point within the complex for generating the new generation, two different jump rates are used. The reason for different jump rates is to allow MFL to have a better exploration and exploitation ability during optimization process. These jump rates are selected by trial and error and may need further investigation to achieve a better performance by MFL algorithm. Fourth, when the generated offspring is not better than the parents, a new point is randomly selected within the range of individuals in the subcomplex. This process, which is referred to as censorship step in the FL algorithm (Eusuff et al. 2006), is different from the original algorithm. The MFL algorithm uses the range of points in the complex rather than the whole feasible parameters range. Resampling within the whole parameter space can decrease the convergence speed of the FL. Hence, the resampling process is carried out only within the range of points in the complex. Hereafter, the SC-SAHEL with MFL algorithm as the only search core is referred as SC-MFL.

2.2.3 Modified Grey Wolf Optimizer (MGWO)

The Grey Wolf Optimizer is a meta-heuristic algorithm inspired by the social hierarchy and hunting behavior of grey wolves (Mirjalili et al. 2014, Mirjalili et al. 2016). The Grey wolves hunting strategy has three main steps: first, chasing and approaching the prey; second, encircling and pursuing the prey, and finally attacking the prey (Mirjalili et al. 2014). The GWO process resembles the hunting strategy of the Grey wolves. In this algorithm, the top three fittest individuals are selected and contribute to the evolution of population. Hence, the individuals in the population are navigated toward the best solution. The GWO algorithm has shown to be effective and efficient in many test functions and engineering problems. Furthermore, performance of the GWO is comparable to other popular optimization algorithms, such as GA and PSO (Mirjalili et al. 2014). GWO follows an adaptive process to update the jump rates, to maintain balance between exploration and exploitation phases. The adaptive jump rate of the GWO is removed here and 3 different jump rates are used instead. The reason for this modification is that the information given to each EA is limited to its assigned complex. Similar to MFL algorithm, the modified GWO (MGWO) algorithm uses the range of parameters to resample individuals, when the generated offspring are not superior to their parents. Details and pseudo code of the MGWO algorithm can be found in the Appendix C. Hereafter, the SC-SAHEL with MGWO algorithm as the only search core is referred as SC-MGWO.

2.2.4 Differential Evolution (DE)

The DE algorithm is a powerful but simple heuristic population-based optimization algorithm (Omran et al. 2005, Sadegh and Vrugt 2014) proposed by Storn and Price (1997). In 2011, Mariani et al. (2011) integrated the DE algorithm into SCE-UA framework and showed that the new framework is able to provide more robust solutions for some optimization problems in comparison to the SCE-UA. Similar to the work by Mariani et al. (2011), we use a slightly modified DE algorithm based on the concepts from Omran et al. (2005), in order to integrate the DE algorithm into the SC-SAHEL framework. As the DE algorithm has slower performance in comparison to other EAs used here, we have added multiple steps to the DE. Here, the DE algorithm uses three different mutation rates in three attempts. In the first attempt, the algorithm uses a larger mutation rate. This helps exploring the search space with larger jump rates. In the second attempt, the algorithm reduces the mutation rate to a quarter of the first attempt. This will enhance the exploitation capability of the EA. If none of these mutation rates could generate a better offspring than the parents, in the next attempt the mutation rate is set to half of the first attempt. Lastly, if none of these attempts generate a better offspring in comparison to the parents, a new point is randomly selected within the range of individuals in the complex. The pseudo code of the modified DE algorithm is detailed in Appendix D. The SC-SAHEL algorithm is referred to as SC-DE, when the DE algorithm is used as the only search algorithm.

3 Conceptual test functions and results

3.1 Test functions

The SC-SAHEL framework is benchmarked over 29 mathematical test functions using single-method and multi-method search mechanisms. This includes 23 classic test functions obtained from Xin et al. (1999). The name and formulation of these functions along with their dimensionality and range of parameters are listed in Table 1. We selected these test functions as they are standard and popular benchmarks for evaluating new optimization algorithms (Mirjalili et al. 2014). The remaining 6 are composite test functions, cf_{1-6}, (Liang et al. 2005), which represent complex optimization problems. Details of the composite test functions can be found in the work of Liang et al. (2005) and Mirjalili et al. (2014). Classic test functions have dimensions in the range of 2 to 30, and all the composite test functions are 10 dimensional. Figures 2 and 3 show response surface of these test functions in 2-dimension form. The SC-SAHEL settings used for optimizing these test functions are listed in Table 2 for each test function. Number of points in each complex and number of evolution steps for each complex are set to $2d+1$ and $\max(d+1,10)$, respectively, where d is the dimension of the problem. The number of evolution steps is set to $\max(d+1,10)$, to guarantee that EAs evolve the complexes for enough number of steps, before evaluating the EAs. In the high-dimension problems, the maximum number of function evaluation should be selected with careful consideration.

Several experiments were conducted to find an optimal set of parameters for the SC-SAHEL setting. These experiments revealed that a low number of evolutionary steps before shuffling the complexes, may not show the potential of the EAs. On the other hand, using a large value for the number of evolution steps may shrink the complex to a small space, which cannot span the whole search space (Duan et al. 1994). Maximum number of function evaluation is determined according to the complexity of the problem and is different for each of the test cases. In addition to the maximum number of function evaluation, the range of the parameters in the population and the improvement in the objective function values are used as convergence criteria. The optimization run is terminated if the population range is smaller than 10^{-7}% of the feasible range or the improvement in (objective) function value is smaller than 0.1% of the mean (objective) function value in the last 50 shuffling steps. The LHS mechanism is used as the sampling algorithm of SC-SAHEL for generating the initial population. The framework provides multiple settings for boundary handling, which can be selected by the user. SC-SAHEL uses reflection as the default boundary handling method. Other initial sampling and boundary handling methods are also implemented in the SC-SAHEL framework. Sensitivity of the initial sampling and boundary handling on the performance of the SC-SAHEL algorithm is not studied in this paper. The aforementioned settings can be applied to a wide range of problems.

Table 1. The detailed information of 23 test functions from Xin et al. (1999), including mathematical expression, dimension, parameters range and global optimum value (f_{min}).

Function Number	Name	Function	Dim	Range	f_{min}
$f_1(x)$	Sphere Model	$f(x) = \sum_{i=1}^{n} x_i^2$	30	[-100,100]	0
$f_2(x)$	Schwefel's Problem 2.22	$f(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	[-10,10]	0
$f_3(x)$	Schwefel's Problem 1.2	$f(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$	30	[-100,100]	0
$f_4(x)$	Schwefel's Problem 2.21	$f(x) = max_i\{\lvert x_i \rvert, 1 \leq i \leq n\}$	30	[-100,100]	0
$f_5(x)$	Generalized Rosenbrock's Function	$f(x) = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right]$	30	[-30,30]	0
$f_6(x)$	Step Function	$f(x) = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$	30	[-100,100]	0
$f_7(x)$	Quartic Function	$f(x) = \sum_{i=1}^{n} ix_i^4 + random[0,1)$	30	[-1.28,1.28]	0
$f_8(x)$	Generalized Schwefel's Problem 2.26	$f(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	30	[-500,500]	-12569.5
$f_9(x)$	Generalized Rastrigin's Function	$f(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	30	[-5.12,5.12]	0
$f_{10}(x)$	Ackley's Function	$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	30	[-32,32]	0
$f_{11}(x)$	Generalized Griewank Function	$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	[-600,600]	0
$f_{12}(x)$	Generalized Penalized Functions	$f(x) = \frac{\pi}{n}\left\{10\sin^2(\pi y_i) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\right\}$ $+ \sum_{i=1}^{n} u(x_i, 10, 100, 4),$ $y_i = 1 + \frac{1}{4}(x_i + 1),$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \leq x_i \leq a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$	30	[-50,50]	0
$f_{13}(x)$	Generalized Penalized Functions	$f(x) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2[1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\right\}$ $+ \sum_{i=1}^{n} u(x_i, 5, 100, 4)$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \leq x_i \leq a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$	30	[-50,50]	0
$f_{14}(x)$	Shekel's Foxholes Function	$f(x) = \left[\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}\left(x_i - a_{ij}\right)^6}\right]$	2	[-65.536,65.536]	1
$f_{15}(x)$	Kowalik's Function	$f(x) = \sum_{i=1}^{11} \left[a_i - \frac{x_1\left(b_i^2 + b_i x_2\right)}{b_i^2 + b_i x_3 + x_4}\right]^2$	4	[-5,5]	0.0003075
$f_{16}(x)$	Six-Hump Camel-Back Function	$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1 x_2 - 4x_2^2 + 4x_2^4$	2	[-5,5]	-1.0316285
$f_{17}(x)$	Branin Function	$f(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$	2	[-5,10]×[0,15]	0.398
$f_{18}(x)$	Goldstein-Price Function	$f(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]$ $\times\ [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	[-2,2]	3
$f_{19}(x)$	Hartman's Family	$f(x) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{4} a_{ij}\left(x_j - p_{ij}\right)^2\right]$	4	[0,1]	-3.86
$f_{20}(x)$	Hartman's Family	$f(x) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{6} a_{ij}\left(x_j - p_{ij}\right)^2\right]$	6	[0,1]	-3.32
$f_{21}(x)$	Shekel's Family	$f(x) = -\sum_{i=1}^{5} [(x - a_i)(x - a_i)^T + c_i]^{-1}$	4	[0,10]	-10.1532
$f_{22}(x)$	Shekel's Family	$f(x) = -\sum_{i=1}^{7} [(x - a_i)(x - a_i)^T + c_i]^{-1}$	4	[0,10]	-10.4028
$f_{23}(x)$	Shekel's Family	$f(x) = -\sum_{i=1}^{10} [(x - a_i)(x - a_i)^T + c_i]^{-1}$	4	[0,10]	-10.5363

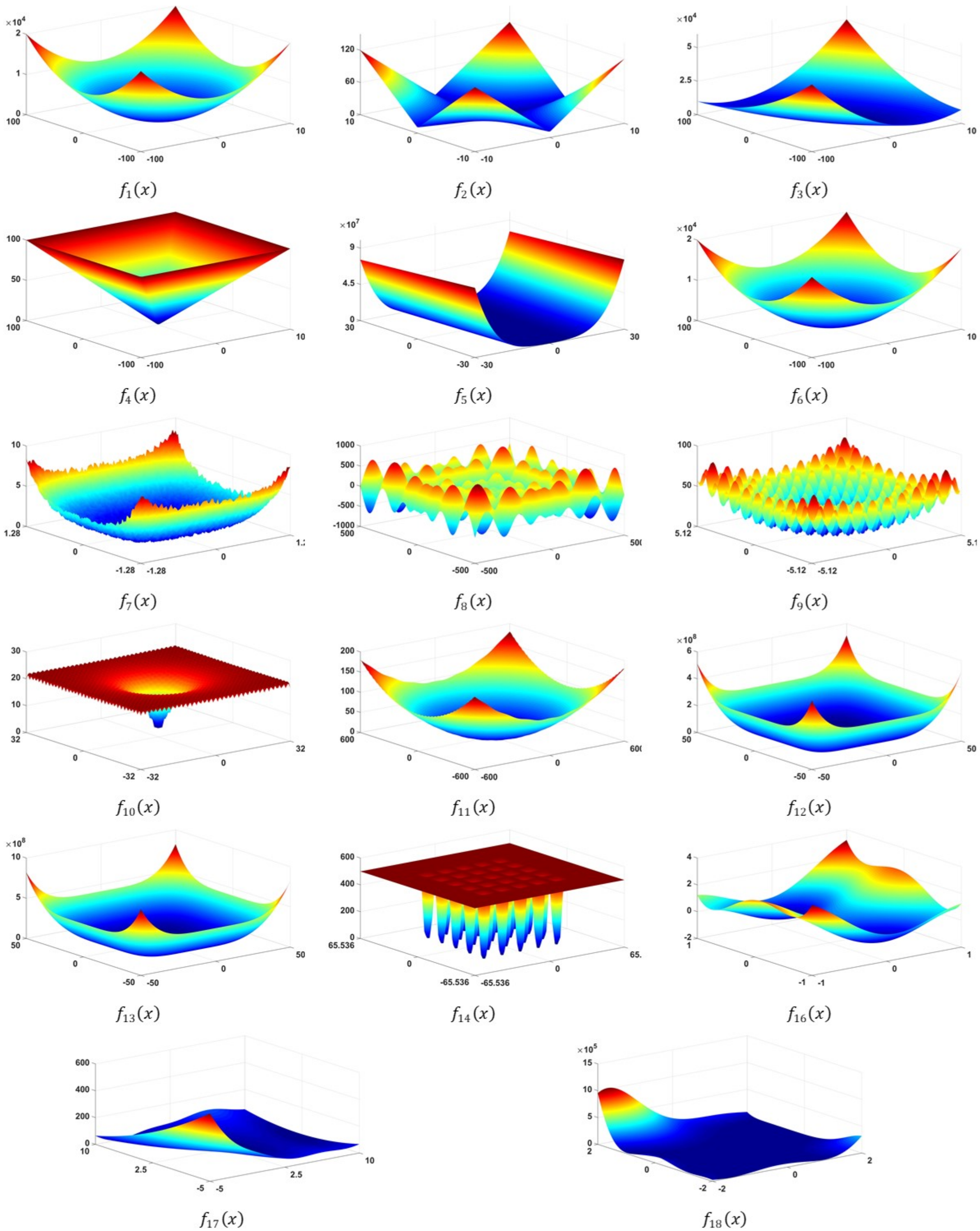

Figure 2. Classic test functions in 2-dimension form

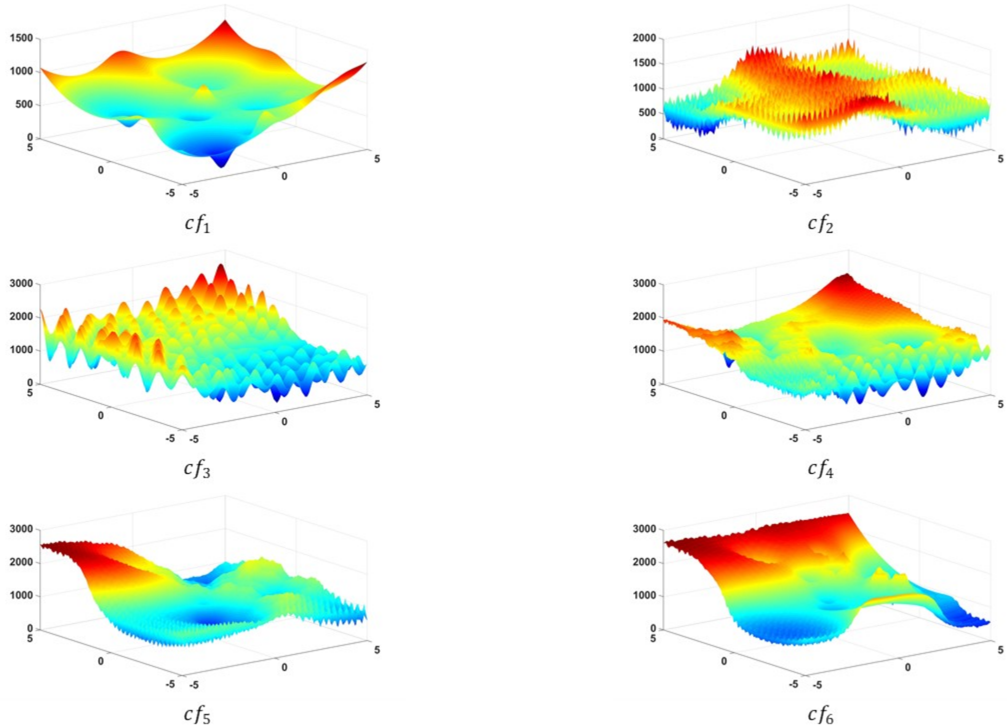

Figure 3. Composite test functions in 2-dimension form

Table 2. List of the settings for the SC-SAHEL algorithm for classic and composite test functions. NGS is the number of complexes, NPS denotes the number of points in each complex and I is the maximum number of function evaluation.

Function	NGS	NPS	I
f_1	8	61	100,000
f_2	8	61	100,000
f_3	8	61	300,000
f_4	8	61	300,000
f_5	8	61	500,000
f_6	8	61	100,000
f_7	8	61	200,000
f_8	8	61	200,000
f_9	8	61	200,000
f_{10}	8	61	200,000
f_{11}	8	61	200,000
f_{12}	8	61	300,000
f_{13}	8	61	400,000
f_{14}	8	10	100,000
f_{15}	8	10	100,000
f_{16}	8	10	100,000
f_{17}	8	10	100,000
f_{18}	8	10	100,000
f_{19}	8	10	100,000
f_{20}	8	13	100,000
f_{21}	8	10	100,000
f_{22}	8	10	100,000
f_{23}	8	10	100,000
cf_1	16	61	100,000
cf_2	16	61	100,000
cf_3	16	61	100,000
cf_4	16	61	100,000
cf_5	16	61	100,000
cf_6	16	61	100,000

3.2 Results and Discussion

Table 3 illustrates the statistics of the final function values at 30 independent runs on 29 test functions using the hybrid SC-SAHEL and individual EAs, with the goal to minimize the function values. The best mean function value obtained for each test function is expressed in bold in Table 3. Results show that the hybrid SC-SAHEL achieved the lowest function values in 15 out of 29 test functions, compared to the mean function values achieved by all individual algorithms. It is noteworthy that in 20 out of 29 test functions, the hybrid SC-SAHEL was among the top two optimization methods in finding the minimum function value. A two-sample t-test (with 5% significance level) also showed that the result generated with the SC-SAHEL algorithm is generally similar to the best performing algorithms. Comparing among single-method algorithms, in general, the statistics obtained by SP-UCI are superior to other participating EAs. In 12 out of 29 test functions, the SP-UCI algorithm achieved the lowest function value. SC-MFL, SC-MGWO, and SC-DE were superior to other algorithms in 10, 11, and 6 out of 29 test functions, respectively. In test functions f_6, f_{16}, f_{17}, f_{18}, f_{19}, f_{20}, and f_{23}, the single-method and multi-method algorithms achieved same function values on average in most cases. In these cases, according to the statistics shown in Table 3, the SP-UCI and SC-SAHEL algorithms offer lower standard deviation values and show more consistent results as compared to other EAs. The low standard deviation values obtained by SP-UCI and SC-SAHEL indicate the robustness and consistency of these two algorithms in comparison to other algorithms.

Table 3. The mean and Standard deviation (Std) of objective function values for 30 independent runs on 29 test functions using the SC-SAHEL algorithm with single-method and multi-method search mechanism.

Function	SC-SAHEL (MCCE, MFL, MGWO, DE)		SP-UCI (SC-MCCE)		SC-MFL		SC-MGWO		SC-DE	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
f_1	3.68E-11	1.60E-11	**1.68E-11**	1.18E-11	4.29E-11	1.01E-11	5.92E-05	5.51E-05	2.13E-06	2.98E-06
f_2	3.14E-06	3.92E-07	3.00E-06	5.94E-07	**2.35E-06**	2.75E-07	4.12E-03	1.27E-03	6.38E-04	5.26E-04
f_3	**2.11E-10**	6.08E-11	8.95E-10	4.37E-10	4.50E-10	9.15E-11	1.22E+03	2.16E+03	1.86E-09	1.48E-09
f_4	4.89E-06	7.88E-07	8.98E-05	4.60E-05	**3.65E-06**	5.43E-07	5.26E-06	5.59E-07	3.50E-01	2.35E-01
f_5	**7.81E-09**	3.15E-09	2.54E-08	1.52E-08	2.58E+01	2.85E-01	1.28E+01	1.85	1.33	1.91
f_6	**0**	0	**0**	0	**0**	0	3.33E-02	1.83E-01	6.33E-01	6.69E-01
f_7	1.09E-03	5.33E-04	**4.78E-04**	3.44E-04	1.37E-03	6.36E-04	1.34E-02	4.90E-03	2.08E-03	8.93E-04
f_8	**-9.87E+03**	6.14E+02	-5.09E+03	2.27E+02	-4.36E+03	2.90E+02	-4.91E+03	3.75E+02	-9.75E+03	6.41E+02
f_9	8.29E-01	1.73	**3.32E-02**	1.82E-01	1.60E+01	9.78	2.01E+02	1.19E+01	2.67E+01	4.57E+01
f_{10}	1.49E-06	2.43E-07	**1.08E-06**	2.55E-07	1.52E-06	2.00E-07	5.47E-06	5.34E-07	1.42	4.98E-01
f_{11}	**8.05E-11**	2.08E-11	1.77E-10	5.19E-11	1.61E-04	8.81E-04	7.21E-03	1.15E-02	1.42E-02	1.51E-02
f_{12}	**1.58E-13**	5.02E-14	5.27E-13	3.38E-13	1.31E-01	8.80E-02	1.06E-12	1.80E-13	3.11E-02	7.77E-02
f_{13}	3.66E-04	2.01E-03	**2.55E-12**	8.69E-13	7.15E-02	8.94E-03	1.62E-11	3.31E-12	3.97E-03	6.59E-03
f_{14}	**9.98E-01**	1.40E-16	**9.98E-01**	1.27E-16	2.53	3.13	**9.98E-01**	2.16E-16	1.99	1.51
f_{15}	**3.07E-04**	5.61E-17	1.19E-03	3.80E-03	1.08E-03	3.68E-03	**3.07E-04**	8.87E-14	2.98E-03	6.93E-03
f_{16}	**-1.03**	1.37E-15	**-1.03**	7.61E-16	**-1.03**	6.28E-07	**-1.03**	9.51E-15	**-1.03**	1.18E-15
f_{17}	**3.98E-01**	1.47E-15	**3.98E-01**	0	**3.98E-01**	2.05E-04	**3.98E-01**	7.63E-15	**3.98E-01**	0.00
f_{18}	**3.00**	2.20E-14	**3.00**	1.25E-14	**3.00**	1.81E-05	**3.00**	7.30E-14	**3.00**	1.72E-14
f_{19}	**-3.86**	2.08E-15	**-3.86**	2.12E-15	**-3.86**	5.46E-05	**-3.86**	1.61E-15	**-3.86**	1.97E-15
f_{20}	**-3.32**	2.17E-02	**-3.32**	2.17E-02	-3.31	3.03E-02	-3.25	5.92E-02	-3.31	4.11E-02
f_{21}	-9.16	2.58	-5.92	3.28	**-9.69**	1.75	-9.48	1.75	-8.97	2.18
f_{22}	-1.02E+01	9.63E-01	-9.64	2.31	**-1.04E+01**	4.56E-04	**-1.04E+01**	5.05E-13	-9.35	2.46
f_{23}	**-1.05E+01**	1.93E-13	-1.03E+01	1.22	**-1.05E+01**	6.96E-06	**-1.05E+01**	5.00E-13	-9.64	2.35
cf_1	6.67	2.54E+01	3.33	1.83E+01	1.00E+01	3.05E+01	**9.41E-12**	3.42E-12	1.35E-11	5.66E-12
cf_2	**2.00E+01**	4.84E+01	1.23E+02	6.79E+01	7.76E+01	4.59E+01	3.94E+01	1.44E+01	3.14E+01	5.39E+01
cf_3	1.32E+02	9.33E+01	1.33E+02	8.22E+01	2.80E+02	3.16E+01	3.00E+02	4.21E+01	**1.28E+02**	3.83E+01
cf_4	2.71E+02	6.67E+01	2.93E+02	8.38E+01	3.46E+02	1.47E+01	3.30E+02	4.15E+01	**2.63E+02**	3.20E+01
cf_5	1.70E+01	3.77E+01	9.75E+01	1.83E+01	3.05E+01	4.33E+01	**3.37**	1.83E+01	1.10E+01	3.05E+01
cf_6	6.71E+02	2.00E+02	8.72E+02	6.59E+01	7.80E+02	1.85E+02	**5.40E+02**	1.23E+02	6.38E+02	1.86E+02

In the test functions that the hybrid SC-SAHEL algorithm was not able to produce the best mean function value, the achieved mean function values deviation from that of the best-performing algorithms are marginal. For instance, on the test functions f_2, f_4, f_{10}, and f_{22}, the statistics of the values obtained by SC-SAHEL are similar to that achieved by the best-performing methods, which are SP-UCI, and SC-MFL, respectively. In general, the hybrid SC-SAHEL algorithm is superior to algorithms with individual EA on most of the test functions, although on some test functions, the SC-SAHEL algorithm is slightly inferior to the best-performing algorithm with only marginal differences. The performance of the SC-SAHEL in these test functions can be attributed to two main reasons. First, in the hybrid algorithm, all the EAs are involved in the evolution of the population. Hence, if one of the algorithms have poor performance in comparison to other EAs, it still evolves a portion of the population. As the complexes are evolved independently, the poor-performing EAs may devastate a part of the information in the evolving complex. On the other hand, when the algorithms are used individually in the SC-SAHEL framework, the EA utilizes the information in all the complexes and the whole population. In this case, better result will be achieved in comparison to the hybrid SC-SAHEL, if the EA is the fittest algorithm for the problem space. Second, some of the EAs are faster and more efficient in a specific optimization phase (exploration/exploitation) than others. However, they might not be as effective as other EAs for other optimization phases. Hence, dominance of these algorithm during the exploration or exploitation phases can mislead other EAs and cause early (and premature) convergence. Engagement of other algorithms in the evolution process may prevent early convergence in these cases. Generally, the performance criteria, EMP, is responsible for selecting the most suitable algorithm in each optimization step, however, the criteria used in the SC-SAHEL is not guaranteed to perform well in all problem spaces. The performance criteria are problem dependent and need further investigations based on the problem space and EAs. However, the EMP metric seems to be a suitable metric for a wide range of problems.

To further evaluate the performance of the hybrid SC-SAHEL algorithm, we present the success rate of the algorithms in Figure 4. The success rate is defined by setting target values for the function value for each test function. When the function value is smaller than the target value, the goal of optimization is reached, and therefore, the algorithm is considered successful. A higher success rate resembles a better performance. We use same target value for all algorithms in order to have a fair comparison. According to Figure 4, in 16 out of 29 test functions, the hybrid algorithm achieved 100% success rate. In other cases, the success rates achieved by the proposed hybrid algorithm are comparable to the best-performing algorithm with single EA. For instance, on the test function f_9, the SC-MGWO, SC-DE and SC-MFL are not successful in finding the optimum solution (success rates are 0%, 0%, and 10%, respectively). However, the hybrid SC-SAHEL algorithm has similar performance (80% success rate) to SP-UCI (97% success rate). On the test function f_{21}, the success rate of the hybrid SC-SAHEL algorithm (87%) is close to the SC-MGWO (93%), which is the most successful algorithm. The hybrid SC-SAHEL algorithm also achieved a higher success rate than SP-UCI algorithm (33%) in this test function. According to Figure 4, the average success rate of SC-SAHEL is about 80% over all 29 test functions, and it is the highest compared to the average success rate of other EAs, i.e., 73%, 58%, 58%, and 54% for SP-UCI, SC-MFL, SC-MGWO, and SC-DE algorithm, respectively.

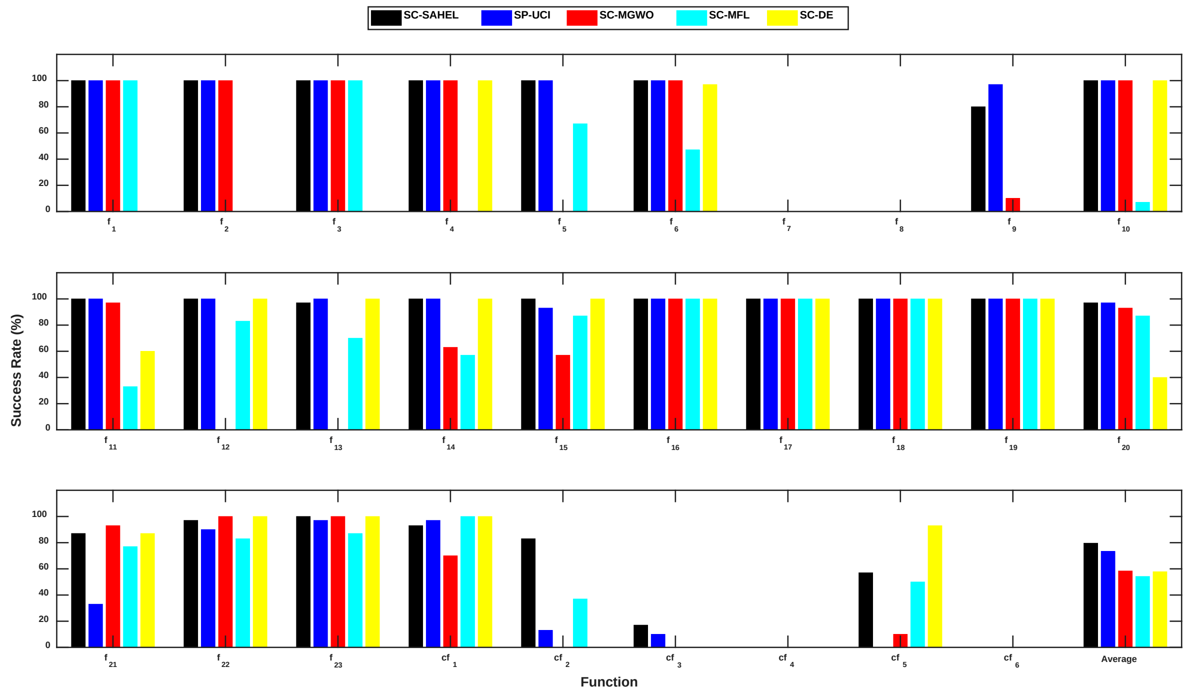

Figure 4. The success rate of the SC-SAHEL using multi-method and single-method search mechanism for 30 independent runs for 29 test functions

In some situations, the poor performing EAs may mislead other EAs and cause early (and premature) convergence. For instance, on the test function cf_5, the hybrid algorithm achieved 57% success rate, which is still better success rate than SP-UCI, SC-MFL and SC-MGWO, which are 0%, 10%, and 50%, respectively. On this test function (cf_5), the performance of the hybrid SC-SAHEL is less affected by the most successful algorithm (DE). This may be due to the low evolution speed of the DE algorithm, as the SC-SAHEL algorithm maintains both convergence speed and efficiency during the entire search. The hybrid SC-SAHEL presents promising performance on the test functions cf_2 and cf_3. On test functions cf_2 and cf_3, the success rate of hybrid SC-SAHEL is significantly higher than other EAs, most of which have 0% success rates. For test function cf_2, the SC-DE algorithm achieved the lowest objective function value and the highest success rate (37%) among single-method algorithms. However, when EAs are combined in the hybrid form, the objective function value and the success rate are significantly improved.

This shows that SC-SAHEL has the capability of solving complex problems by utilizing the potentials and advantages of all participating algorithms and improving the search success rate.

Table 4. The mean and Standard deviation (Std) of the number of function evaluation for 30 independent runs for 29 test functions using the SC-SAHEL algorithm with single-method and multi-method search mechanism.

Function	SC-SAHEL (MCCE, MFL, MGWO, DE)		SP-UCI (SC-MCCE)		SC-MFL		SC-MGWO		SC-DE	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
f_1	32816.33	723.0532	**26877.93**	609.7676	100199.9	129.6894	33012.97	284.4416	100325.5	144.6016
f_2	39298.4	917.7627	**29333.33**	674.156	100193.7	126.3259	35876.77	344.1018	100307.9	168.2884
f_3	91746.23	2288.806	**73474.5**	5435.367	226848.3	20135.41	239199.9	31109.82	241449.1	90118.79
f_4	50197.6	1761.589	82183.67	19213.58	300252.9	121.2331	**37987.4**	1170.002	227316.4	5488.107
f_5	335364.2	8102.236	401124.8	14599.33	439093.2	47901.23	**118900.7**	38671.69	500310.9	163.5498
f_6	40293.63	377.0177	**32537.93**	151.8836	66102.9	3708.332	43063.63	1463.074	90205.23	5773.92
f_7	**69779.5**	23763.53	69823.27	24314.74	78895.43	24205.89	81421.53	22877.8	117468.4	33083.6
f_8	71834.83	9826.963	54020	17225.97	65629.77	5201.429	**45254.8**	15104.68	62555.83	21579.07
f_9	59710.57	12460.93	**33949.6**	996.5881	100705.8	22607.84	85055.73	20771.06	90930.3	34180.61
f_{10}	33765.77	887.3708	**27116.33**	379.9873	77520.6	15528.44	33181.03	416.0297	165489.4	4726.909
f_{11}	35504.9	629.8192	**30623.53**	860.8274	117357.6	13250.36	38652.4	19330.05	155148.6	16730.48
f_{12}	55908.07	4735.601	**39264.23**	3125.88	141722.1	31245.81	88234.23	28948.91	181820.6	5132.966
f_{13}	54148.7	3949.577	**32262.23**	851.2965	123903.4	20354.81	72334.73	22345.94	170930.5	3295.191
f_{14}	5216.333	443.942	5708.433	764.8273	4829.2	841.1355	14986.77	3537.605	**4530.2**	322.0474
f_{15}	9059.8	551.1741	**6517.167**	2358.518	8144.667	1151.183	66441.3	35455.39	18813.63	659.7369
f_{16}	3700.133	1115.033	**2746.3**	747.3937	3491.933	574.5392	8549.4	1494.076	3490.733	556.9577
f_{17}	3665.533	601.3615	**2910.633**	624.4692	3552.267	538.6385	11453.13	2592.687	5115.367	2082.727
f_{18}	2837.933	308.7723	**2000.633**	151.8512	2899.567	299.5645	8405.933	1320.571	2833.5	111.3877
f_{19}	4225.733	424.8389	**2852.2**	95.83045	4233.4	238.4404	13183.77	519.0798	4983.9	179.5704
f_{20}	8915.833	1069.182	**5645.567**	268.4028	8858.967	300.0987	17143.33	1316.025	12691.67	1818.526
f_{21}	7455.033	1741.525	**7377.533**	3260.208	7471.4	1554.087	18771.33	2996.925	10755.57	1573.865
f_{22}	6370.5	869.9209	**4512.433**	1290.258	7541.433	1582.216	17466.23	834.9485	8728.7	927.622
f_{23}	6200.133	614.6406	**4084.233**	464.9113	6823.7	327.7709	17351.87	861.8541	8398.067	599.0103
cf_1	15049.43	875.8969	**10293.43**	233.9694	21663.47	921.1805	74089.17	17803.2	28321.6	678.283
cf_2	16527.63	1432.21	**10586.8**	464.8359	20285.83	2346.093	36617.1	13118.85	30686.4	9354.096
cf_3	25991.03	8041.928	**16021.5**	3833.203	23801.2	3604.495	19323.13	6052.813	29496.9	9113.814
cf_4	22873.87	4414.168	**16510.13**	4052.642	21121.93	2417.582	23841.93	8026.638	35134.33	14468.31
cf_5	17044.53	1350.845	**13512.2**	746.6731	21400.57	1759.215	53551.43	23577.95	39200.77	4125.908
cf_6	13779.33	2279.744	**10518.1**	2977.194	14967.5	2820.062	22265.8	15340.72	27734.83	4606.317

In Table 4, we present the mean and standard deviation of the number of function evaluation, which indicates the speed of each algorithm. As one of the stopping criteria in SC-SAHEL framework is the maximum number of function evaluation, some algorithms may terminate before they show their full potential. For instance, the SC-DE and the SC-MFL, usually reach the maximum number of function evaluations, while other algorithms satisfy other convergence criteria in much less number of function evaluations. In this case, the objective function value doesn't represent the potential of the slow algorithms. To give a better insight into this matter, the mean and standard deviation (Std) of the number of function evaluations are compared in Table 4. The goal is to compare the speed of the individual EAs and the hybrid optimization algorithm. According to Table 4, in most of the test cases, the SP-UCI algorithm has the least number of function evaluations, regardless of the objective function value achieved by the EAs.

Comparing the success rate and the number of function evaluation for different EAs shows that SP-UCI achieved 100% success rate with the lowest number of function evaluation, in 15 out of 29 test functions. The SC-MGWO algorithm only achieved 100% success rate with the lowest number of function evaluation in one test function. Although the hybrid SC-SAHEL algorithm is not the fastest algorithm, its speed is usually close to the fastest algorithm. This is due to the contribution of different EAs in the evolution process and the EAs behavior on different problem spaces. For instance, DE algorithm is slower in comparison to MCCE (SP-UCI) algorithm in most of the test functions. Hence, when the algorithms are working in a hybrid form, the hybrid algorithm will be slower than the situation when the MCCE (SP-UCI) algorithm is used individually.

Figures 5, 6, and 7 compare the average number of complexes assigned to each EA for the 29 employed test functions during the course of the search. The variation of the number of complexes assigned to each EA indicates the dominance of each EA during the course of the search. Hence, the performance of EAs at each optimization step can be monitored. In many test cases, MCCE (SP-UCI) algorithm has a relatively higher number of complexes than other EAs during the search. This shows that MCCE is a dominant search algorithm on most of the test functions. However, in some other cases, MCCE is only dominant in a certain period of the search, while other EAs have demonstrated better efficiency during the entire search. For example, on test functions f_7 and f_{20}, MCCE algorithm appears to be dominant only during the beginning of the search. In the test function f_7, the exploration process starts with the dominance of the MCCE and shifts between MGWO and MFL after the first 20 shuffling steps. In some of the test functions, such as f_7, a more random fluctuation is observed in the number of complexes assigned to each EA. The reason for this behavior is that EAs have very close competition in these shuffling steps. Due to the noisy response surface of the test function f_7, most of the EAs cannot significantly improve the (objective) function values during the exploitation phase. On test functions f_8 and f_{18}, the MFL and DE algorithms are the dominant, respectively, during the beginning of the run, while MCCE algorithm becomes dominant only when the algorithm is in exploitation phase. Lastly, on test functions f_9, f_{22}, cf_1, and cf_4, the variations of the number of complexes and the precedence of different EAs as the most dominant search algorithm are observed.

It is worth mentioning that, Figures 5, 6, and 7 show the number of complexes assigned to each EA for a single optimization run. Our observation of each individual run results (not shown herein) shows variation of the number of complexes among different runs is similar to each other for most test cases. The observed variation for individual runs follows a specific pattern and is not random. The similarity of the EAs dominance pattern indicates that the selection of the EAs by the SC-SAHEL framework only depends on the characteristics of the problem space and the EAs employed. This also indicates that different EAs have pros and cons on different optimization problems.

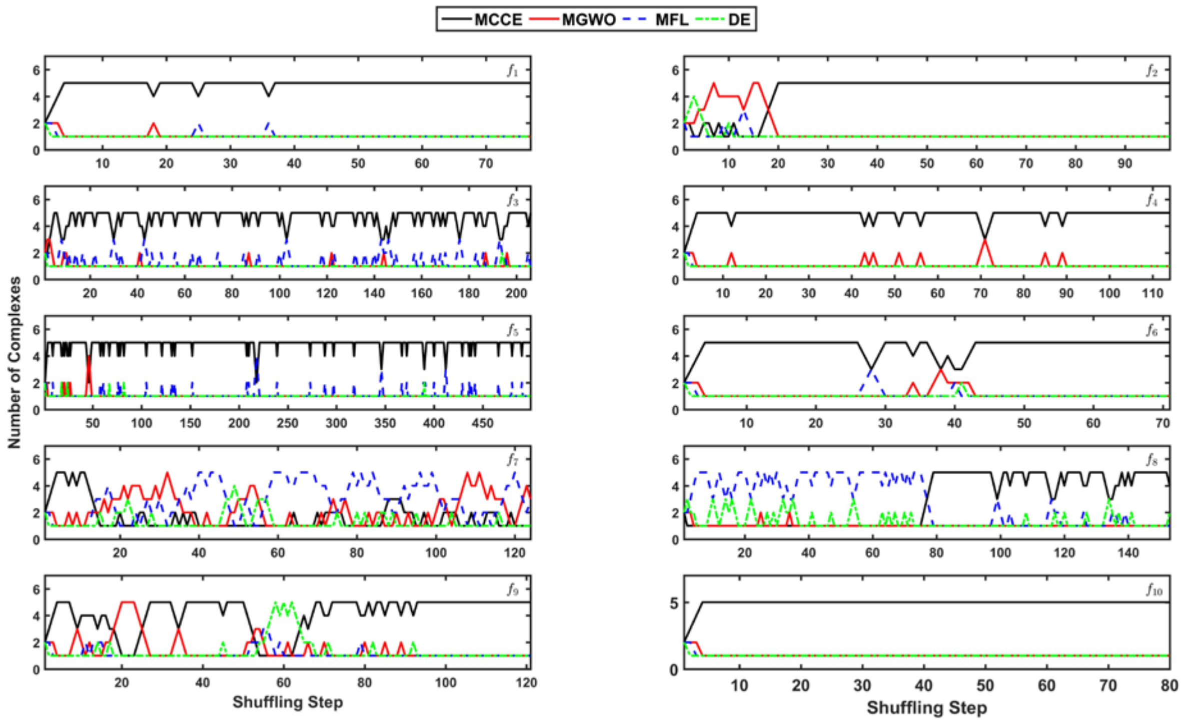

Figure 5. Number of complexes assigned to EAs during the entire optimization process on test function f_1-f_{10}

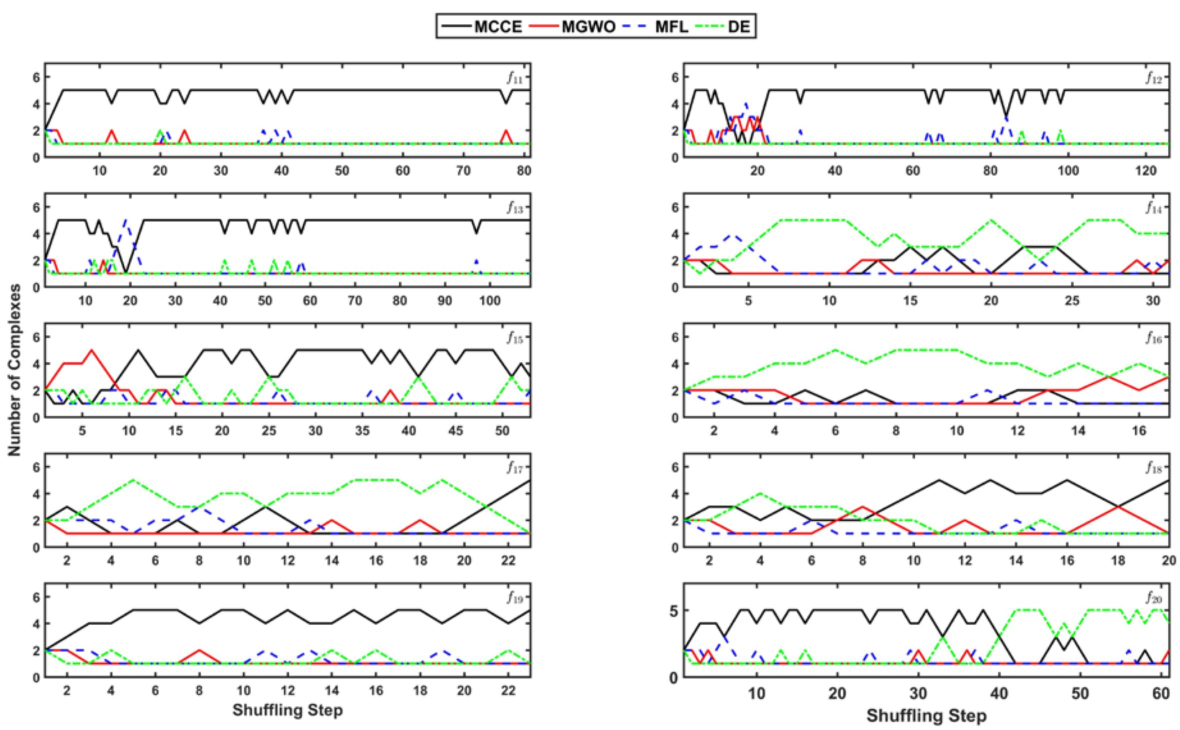

Figure 6. Number of complexes assigned to EAs during the entire optimization process on test function f_{11}-f_{20}

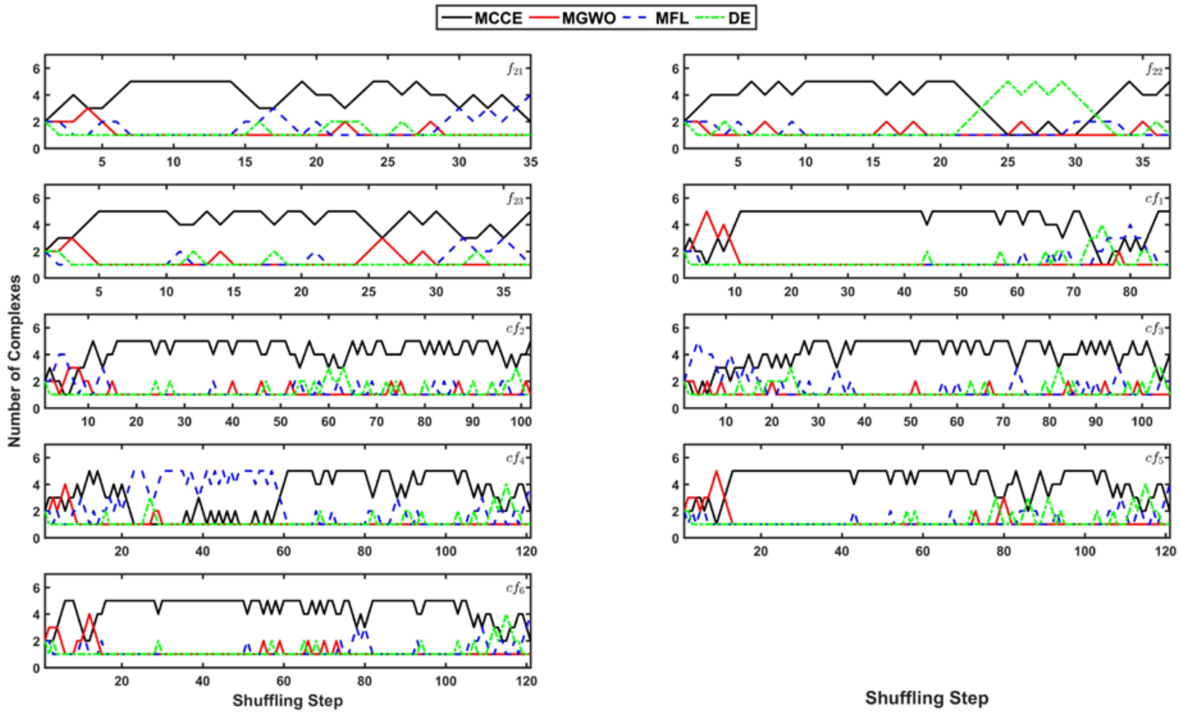

Figure 7. Number of complexes assigned to EAs during the entire optimization process on test function f_{21}-f_{23} and cf_1-cf_6

As a summary of our experiments on the conceptual test functions (Tables 3, and 4, and Figure 4, 5, 6, and 7), the main advantage of the SC-SAHEL algorithm over other optimization methods is its capability of revealing the trade-off among different EAs and illustrating the competition of participating EAs. Different optimization problems have different complexity, which introduces various challenges for each EA. By incorporating different types of EAs in a parallel computing framework, and implementing an "award and punishment" logic, the newly developed SC-SAHEL framework not only provides an effective tool for global optimization but also gives the user insights about advantages and disadvantages of each participating EAs on individual optimization tasks. This shows the potential of the SC-SAHEL framework for solving different class of problems with different level of complexity. Besides, the hybrid SC-SAHEL algorithm is superior to shuffled complex-based methods with single search mechanism, such as SP-UCI, in an absolute majority of the test functions.

4 Example applications and results

In this section, we demonstrate an example application of the newly developed SC-SAHEL algorithm. A conceptual reservoir model is developed with the goal of maximizing hydropower generation on a daily-basis operation. The model is applied to the Folsom reservoir in Northern California.

4.1 Reservoir Model

A conceptual model is set up based on the relationship between the hydropower generation, storage, water head and bathymetry of the Folsom reservoir. Daily releases from the reservoir in the study period are treated as the parameters of the model, which in turn determines the problem dimensionality. The model objective is to maximize the hydropower generation for a specific period. The total hydropower production is a function of the water head difference between forebay and tailwater and the turbine flow rate. The driving equation of the model is based on mass balance (water budget), which is formulated as,

$$S_t = S_{t-1} + I_t - R_t \pm M_t, \tag{2}$$

where S_t is storage at time step t, I_t and R_t signify total inflow and release from the reservoir at time t, respectively. M_t is total outflow/inflow error which is derived by setting up mass balance for daily observed data. The objective function employed here is,

$$\mathrm{OF} = \sum_{t=1}^{N} 1 - \frac{P_t}{P_c}, \tag{3}$$

where P_c is total power plant capacity in MW and P_t is total power generated in day t in MW. For each day P_t is derived as follow,

$$P_t = \eta \rho g Q_t H_t, \tag{4}$$

where η signifies turbine efficiency, ρ is water density (Kg/m^3), g is gravity (9.81 m/s^2) and Q_t is discharge (m^3/s) at time step t. H_t is hydraulic head (m) at time step t, which is defined as,

$$H_t = h_f - h_{tw}, \tag{5}$$

where h_f and h_{tw} are water elevation in forebay and tailwater, respectively. h_f and h_{tw} are derived by fitting a polynomial to reservoir bathymetry data.

In the reservoir model coined above, multiple constraints are considered for better representation of the real behavior of the system. These constraints include power generation capacity, storage level, spill capacity, and changes in the daily hydropower discharge. Total daily power generation is compared to maximum capacity of the hydropower plant. Also, rule curve is used to control reservoir storage level during the operation period. Besides, final simulated reservoir storage is constrained to 0.9 - 1.1 of the observed storage. In another word, 10% variation from the observation data is allowed for the final simulated storage level. This constraint adds information from real reservoir operation into the optimization process. This constraint can be replaced by other operation rules for simulation purposes. The spill capacity of dam is calculated according to the water level in the forebay and compared to simulated spilled water. A quadratic function is fitted to the water level and spill capacity data, to derive the spill capacity at each time step. The change in daily hydropower release is also constrained to better represent actual hydropower discharge and avoid large variation in a daily release.

The reservoir model used here is non-linear and continuous. The constraints of the model render finding the feasible solution a challenging task for all the EAs. The SC-SAHEL framework is used to maximize the hydropower generation by minimizing the objective function value. The settings used for the SC-SAHEL is similar to the settings used for the mathematical test functions. However, the maximum number of function evaluations is set to 10^6. Lower bound of the parameters' range varies monthly due to the operational rules; however, upper bound is determined according to the hydraulic structure of the dam.

4.2 Study basin

Folsom reservoir is located on the American river, in northern California and near Sacramento, California. Folsom dam was built by the US Army Corps of Engineers during 1948 to 1956, and is a multi-purpose facility. The main functions of the facility are flood control, water supply for irrigation, hydropower generation, maintaining environmental flow, water quality purposes, and providing recreational area. The reservoir has a capacity of 1,203,878,290 m^3 and the power plant has a total capacity of 198.7 MW. Three different periods are considered here. The first study period is April 1st, 2010 to June 30th, 2010. The year 2010 is categorized as below-normal period according to California Department of Water Resources. The same period is selected in 2011 and 2015, as former is categorized by California Department of Water Resources as wet, and latter is classified as critical dry year. The input and output from the reservoir are obtained from California Data Exchange Center (CDEC). Note that demand is not included in the model because the demand data was not available from a public data source.

4.3 Results and Discussion

The boxplot of the objective function values is shown in Figure 8 for the Folsom reservoir during the runoff season in 2015, 2010, and 2011, which are dry, below-normal, and wet years, respectively. The presented results are based on 30 independent optimization runs; however, infeasible objective function values are removed. The feasibility of the solution is evaluated according to the objective function values. Due to the large values returned by the penalty function considered for infeasible solutions, such solutions can be distinguished from the feasible solutions. For wet year (2011) case, SC-MGWO, and SC-DE didn’t find a feasible solution in 2, and 4 runs out of 30 independent runs, respectively. The hybrid SC-SAHEL found feasible solutions in all the cases; however, some of these solutions are not global optima. On average, the hybrid SC-SAHEL algorithm is able to achieve the lowest objective function value as compared to other algorithms during dry and below-normal period. During dry and below-normal periods, SC-SAHEL, SP-UCI, and SC-DE show similar performance. In the wet period, the SP-UCI algorithm achieved the lowest objective function value. The SC-SAHEL algorithm ranked second, comparing the mean objective function values. In this period, the results achieved by the SC-DE is also comparable to SC-SAHEL and SP-UCI. The results show that overall, the hybrid SC-SAHEL algorithm has similar or superior performance in comparison to the single-method algorithms. Also, the results achieved by SC-SAHEL and SP-UCI algorithms has less variability in comparison to other algorithms, which show the robustness of these algorithms. The worst performing algorithm is the SC-MGWO, which achieved the least mean objective function value in all the study periods.

In Figure 9, boxplot of the number of function evaluations is presented for successful runs from the 30 independent runs during dry, below-normal and wet period years. Although the SC-MGWO algorithm satisfied convergence criteria in the least number of function evaluation, the SC-MGWO was not successful in achieving the optimum solution in many cases. The SP-UCI algorithm is the second fastest method among all the algorithms. The hybrid SC-SAHEL, SC-MFL, and SC-DE are the slowest algorithm for satisfying the convergence criteria, in almost all cases. The slow performance of the hybrid SC-SAHEL is due to the fact that 2 out of 4 (DE and MFL) participating EAs have very slow performance over the response surface. Figure 10 demonstrates the number of complexes assigned to each EA during the search, which indicates the dominance of the participating algorithms, and the “award and punishment” logic in the reservoir model. As seen in Figure 10, the MGWO algorithm is dominant in the beginning of the search; although, it is not capable of finding the optimum solution in most cases. The reason for the dominance of the MGWO is the speed of the algorithm in exploring the search space. MGWO is superior to other EAs in the beginning of the search, however, after a few iterations, the MCCE algorithm took the precedence and become the dominant algorithm over other EAs. MGWO and DE are less involved in the rest of the optimization process after the initial steps. However, competition between MCCE and MFL continues. Although contribution of MGWO and DE are at minimum in rest of the optimization process, they are utilizing a part of information within the population. This can affect the speed and performance of the SC-SAHEL algorithm. In both the wet and below-normal cases, the hybrid SC-SAHEL algorithm is mostly terminated by reaching the maximum number of function evolution. However, the mean objective function value obtained by the hybrid SC-SAHEL is still superior to most of the algorithms.

The performance of the SC-SAHEL can be affected by the settings of the algorithm. Different settings have been tested and evaluated for the reservoir model. The results show that the number of evolution steps before shuffling can influence the performance of the hybrid SC-SAHEL algorithm. In the current setting, the number of evolution steps within each complex is set to d+1 (d is dimension of the problem). Although this setting seems to provide acceptable performance for a wide range of problems, it may not be the optimum setting for all the problems spaces and EAs. In the reservoir model, as the study period has 91 days, the model evolves each complex for 92 steps. This number of evolution steps allows the algorithms to navigate the complexes toward local solutions and increase the total number of function evaluations without specific gain. Decreasing the number of evolution steps allows the algorithms to communicate more frequently, so they can use the information obtained by other EAs. Here, for demonstrative purposes, the same setting has been applied to all the problems. However, better performance is observed for the hybrid SC-SAHEL algorithm when the number of evolution steps are set to a value smaller than 92. The algorithm is less sensitive to other settings for the reservoir model, however they can still affect the performance of the algorithm.

In Figure 11, we present the simulated storage level for different study periods achieved by different EAs. During the dry period, not only the SC-SAHEL algorithm achieved the lowest objective function value, but also the storage level is higher than the observed storage level in most of the period. This is due to the fact that, power generation is a function of water height, as well as discharge rate. During below-normal period, SC-SAHEL, SP-UCI, and SC-DE algorithms show a similar behavior in terms of the storage level. During wet period, storage level simulated by SP-UCI and SC-SAHEL algorithm is lower than all other algorithms. It is worth noting that, during wet period, SC-SAHEL and SP-UCI algorithms are able to find optimum solution (which objective function value is 0) in some of the runs. However, the simulated storage by these algorithms show some level of uncertainties (Figure 11). This shows equifinality in simulation, which means that same hydropower generation can be achieved by different sets of parameters (Feng et al. 2017). This equifinality can be due to deficiencies in the model structure, or the boundary conditions (Freer et al. 1996). The wet period seems to offer a more complex response surface for the reservoir model. During the wet period, some algorithms, such as SC-DE, are not capable of finding a feasible solution in some of the runs. In this period, the large input volume and the rule curve added more complexity to the optimization problem. In other study periods, the reservoir level is always below the rule curve.

The results of the real-world application show the potential of the newly developed SC-SAHEL framework for solving high dimension problems. In general, the hybrid algorithm was more successful in finding a feasible solution in comparison to single-method algorithms. In some cases, the hybrid SC-SAHEL was terminated due to the large number of function evaluations. However, the performance of the hybrid SC-SAHEL is always comparable to the best performing method. This shows the potential of the SC-SAHEL for solving a broad class of optimization problems. Besides, the framework provides insight into the performance of the algorithms at different steps of the optimization process. This feature of the SC-SAHEL algorithm can aid the user to select the best setting and EA for the problem.

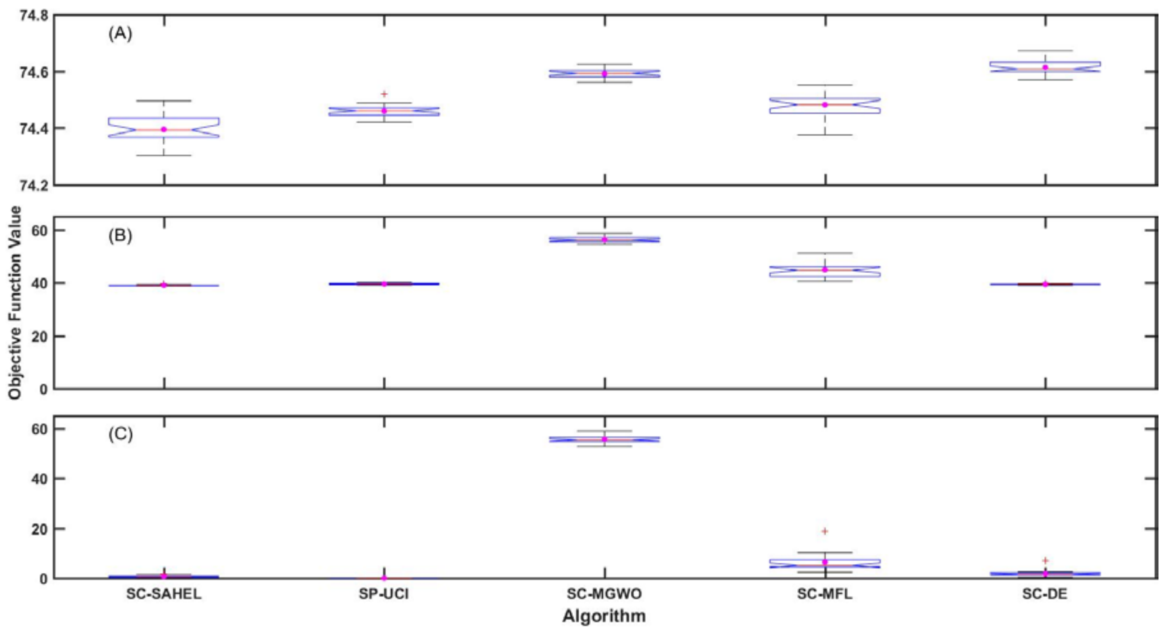

Figure 8. Boxplots of objective function values for successful runs among 30 independent runs, for dry (A), below-normal (B) and wet period (C). The mean of objective functions values is shown with pink marker.

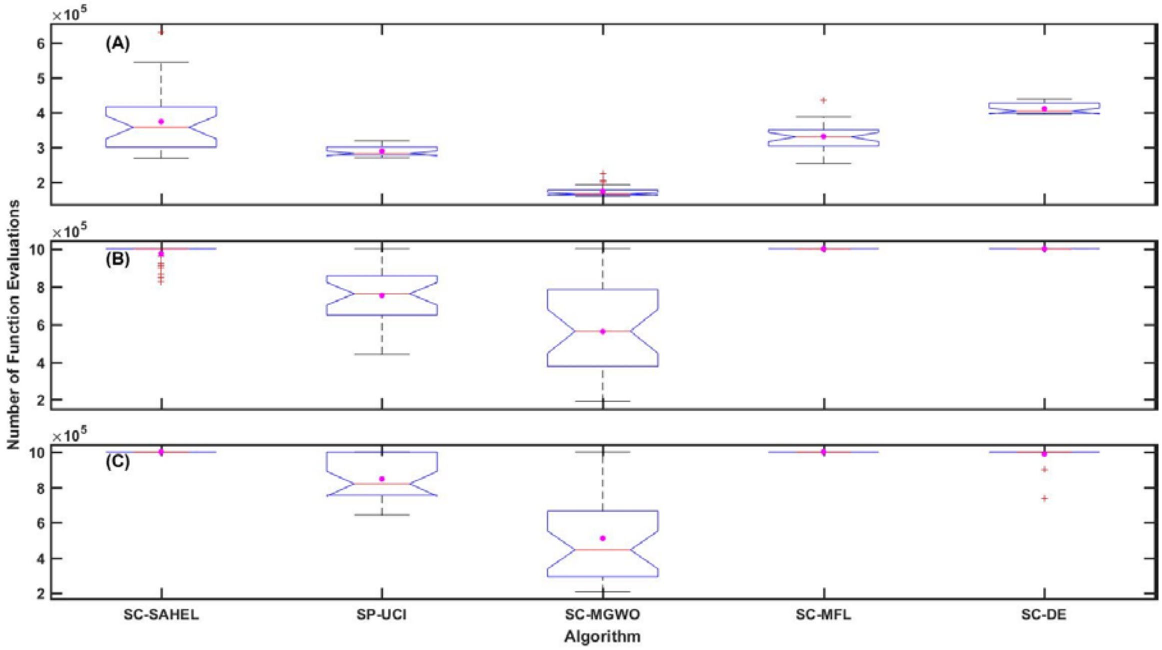

Figure 9 Boxplots of number of function evaluations for successful runs among 30 independent runs for dry (A), below-normal (B) and wet period (C). The mean number of function evaluation is shown with pink marker

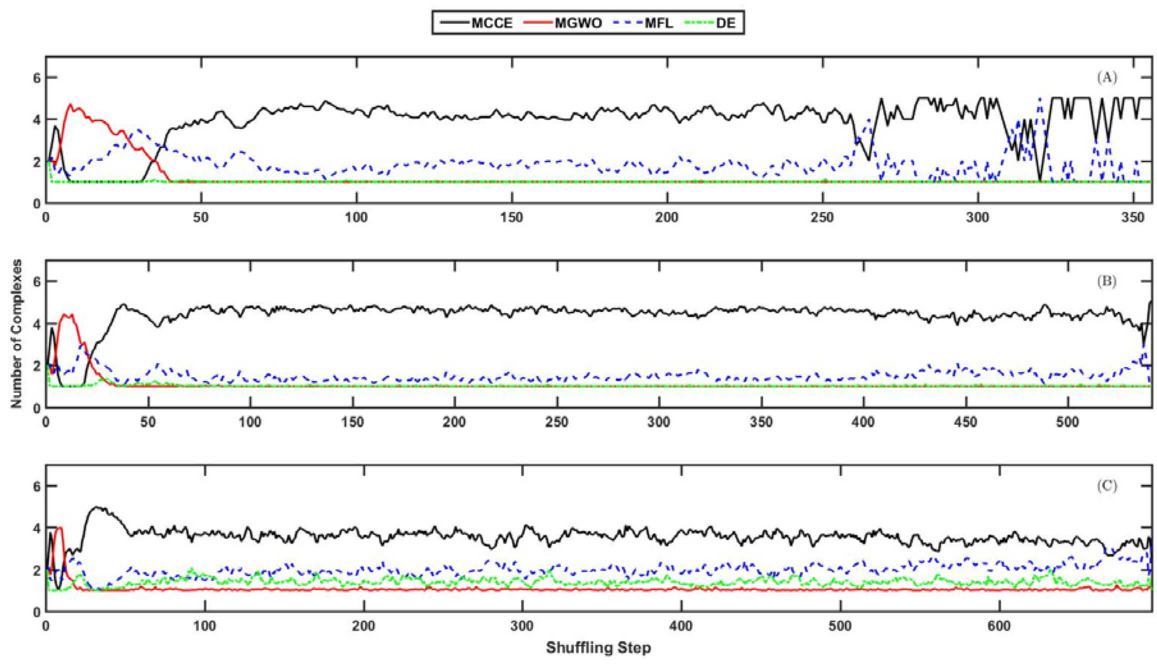

Figure 10. The average number of complexes assigned to each EA at each shuffling step for 30 independent runs for dry (A), below-normal (B), and wet (C) period

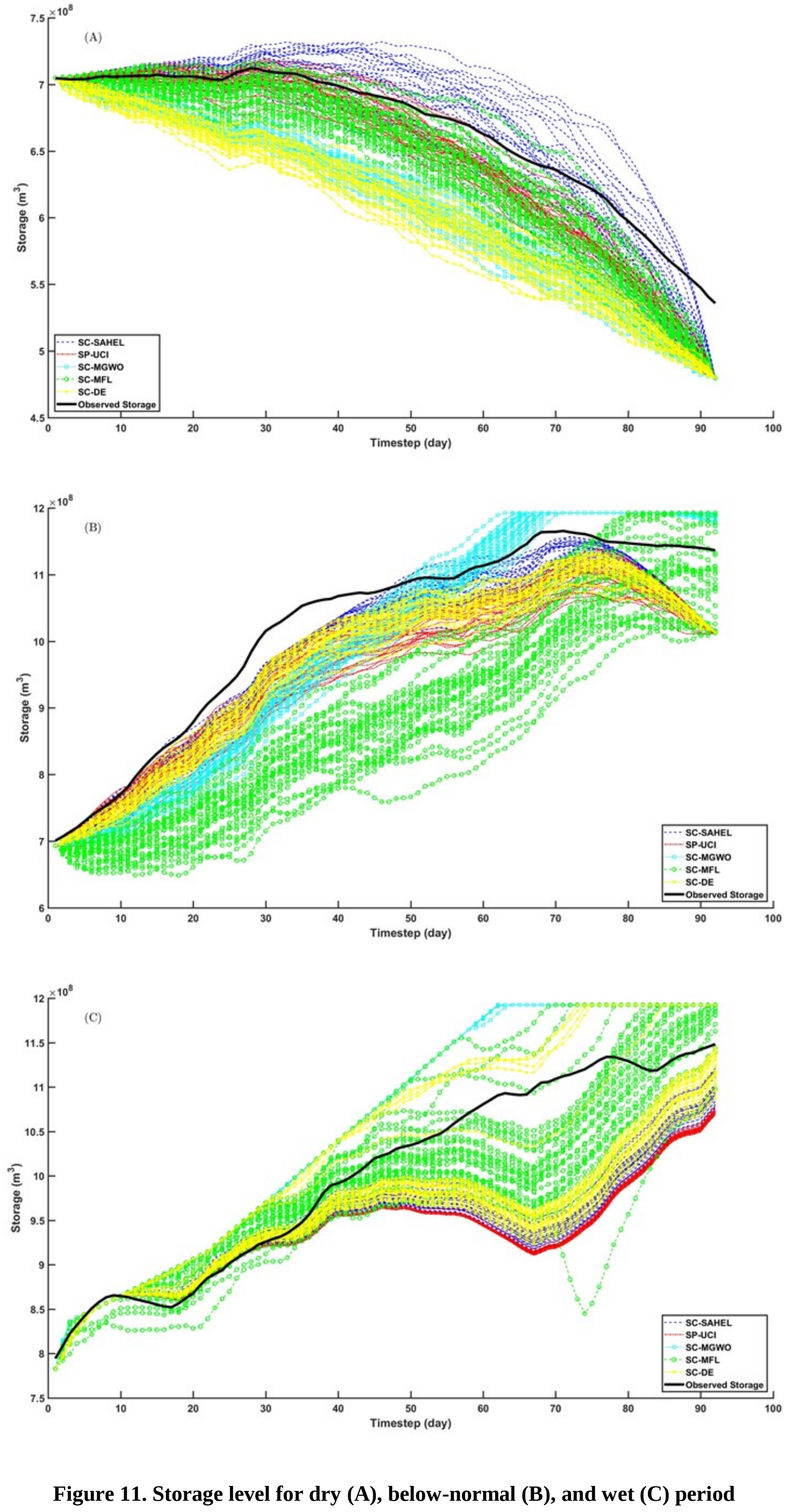

Figure 11. Storage level for dry (A), below-normal (B), and wet (C) period

5 Conclusions and remarks

We developed a hybrid optimization framework, named Shuffled Complex Self Adaptive Hybrid EvoLution (SC-SAHEL), which uses an "award and punishment" logic in junction with various types of Evolutionary Algorithms (EAs), and selects the best EA that fits well to different optimization problems. The framework provides an arsenal of tools for testing, evaluating and developing optimization algorithms. We compared the performance of the hybrid SC-SAHEL with single-method algorithms on 29 test functions. The results showed that the SC-SAHEL algorithm is superior to most of single-method optimization algorithms and in general offers a more robust and efficient algorithm for optimizing various problems. Furthermore, the proposed algorithm is able to reveal the characteristics of different EAs during entire search period. The algorithm is also designed to work in a parallel framework which can take the advantage of available computation resources. The newly developed SC-SAHEL offers different advantages over conventional optimization tools. Some of the SC-SAHEL characteristics are:

- Intelligent evolutionary method adaptation during the optimization process
- Flexibility of the algorithm for using different evolutionary methods
- Flexibility of the algorithm for using initial sampling and boundary handling method
- Independent parallel evolution of complexes
- Population degeneration avoidance using PCA algorithm
- Robust and Fast optimization process
- Evolutionary algorithms comparison for different types of problems

Although the presented results support advantage of the hybrid SC-SAHEL to individual EAs algorithms, there are multiple directions for further improvement of the framework. For example, EAs' performance metric for evaluating the search mechanism. In the current algorithm, the complex allocation to different EA is carried out by ranking the algorithm according to the

EMP metric. The performance criteria can change the allocation process and affect the performance of the algorithm. Depending on the application a more comprehensive performance criterion may be necessary for achieving the best performance. However, the current EMP criterion does not affect the conclusion and comparison of different EAs. In addition, the current SC-SAHEL framework is designed to solve single objective optimization problems. A multi-objective version can be developed to extend the scope of the application. This paper serves as an introduction to the newly developed SC-SAHEL algorithm. We hope that more investigation on the interaction among different EAs, boundary handling schemes and response surface in different case studies and optimization problems reveal the advantages and limitations of SC-SAHEL.

Acknowledgments and data

This work is supported by U.S. Department of Energy (DOE Prime Award # DE-IA0000018), California Energy Commission (CEC Award # 300-15-005), NSF CyberSEES Project (Award CCF-1331915), NOAA/NESDIS/NCDC (Prime award NA09NES4400006 and NCSU CICS and subaward 2009-1380-01), and the U.S. Army Research Office (award W911NF-11-1-0422). The Folsom reservoir bathymetry information used here is provided by Dr. Erfan Goharian from UC Davis, who also helped us for setting up the reservoir model. The authors would like to thank the comments of the fours anonymous reviewers which significantly improved the quality of the paper.

References

Barati, R., Neyshabouri, S.S. and Ahmadi, G. (2014) Sphere drag revisited using shuffled complex evolution algorithm.

Beven, K.J. (2011) Rainfall-runoff modelling: the primer, John Wiley & Sons.

Blum, C. and Roli, A. (2003) Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Comput. Surv. 35(3), 268-308.

Boussaïd, I., Lepagnot, J. and Siarry, P. (2013) A survey on optimization metaheuristics. Information sciences 237(Supplement C), 82-117.

Boyle, D.P., Gupta, H.V., and Sorooshian, S. (2000) Toward improved calibration of hydrologic models: Combining the strengths of manual and automatic methods. Water Resources Research 36(12), 3663-3674.

Chu, W., Gao, X. and Sorooshian, S. (2010) Improving the shuffled complex evolution scheme for optimization of complex nonlinear hydrological systems: Application to the calibration of the Sacramento soil-moisture accounting model. Water Resources Research 46(9), n/a-n/a.

Chu, W., Gao, X. and Sorooshian, S. (2011) A new evolutionary search strategy for global optimization of high-dimensional problems. Information sciences 181(22), 4909-4927.

Coello, C.A.C., Lamont, G.B. and Van Veldhuizen, D.A. (2007) Evolutionary algorithms for solving multi-objective problems, Springer.

Črepinšek, M., Liu, S.-H. and Mernik, M. (2013) Exploration and exploitation in evolutionary algorithms: A survey. ACM Computing Surveys (CSUR) 45(3), 35.

Ding, Y., Wang, C., Chaos, M., Chen, R. and Lu, S. (2016) Estimation of beech pyrolysis kinetic parameters by Shuffled Complex Evolution. Bioresource Technology 200, 658-665.

Dorigo, M., Maniezzo, V. and Colorni, A. (1996) Ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 26(1), 29-41.

Duan, Q., Sorooshian, S. and Gupta, V. (1992) Effective and efficient global optimization for conceptual rainfall-runoff models. Water Resources Research 28(4), 1015-1031.

Duan, Q.Y., Gupta, V.K. and Sorooshian, S. (1993) Shuffled complex evolution approach for effective and efficient global minimization. Journal of Optimization Theory and Applications 76(3), 501-521.

Duan Q., Sorooshian S., Gupta V.K. Optimal use of the SCE-UA global optimization method for calibrating watershed models. Journal of hydrology. 1994 Jun 15;158(3-4):265-84.

Eckhardt, K. and Arnold, J.G. (2001) Automatic calibration of a distributed catchment model. Journal of Hydrology 251(1–2), 103-109.

Erol, O.K. and Eksin, I. (2006) A new optimization method: big bang–big crunch. Advances in Engineering Software 37(2), 106-111.

Eusuff, M., Lansey, K. and Pasha, F. (2006) Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization. Engineering Optimization 38(2), 129-154.

Eusuff, M.M. and Lansey, K.E. (2003) Optimization of water distribution network design using the shuffled frog leaping algorithm. Journal of Water Resources Planning and Management 129(3), 210-225.

Feng, M., Liu, P., Guo, S., Shi, L., Deng, C. and Ming, B. (2017) Deriving adaptive operating rules of hydropower reservoirs using time-varying parameters generated by the EnKF. Water Resources Research 53(8), 6885-6907.

Field, R. and Lund, J.R. (2006) Operating Reservoirs in Changing Conditions, pp. 205-214.

Freer, J., Beven, K. and Ambroise, B. (1996) Bayesian Estimation of Uncertainty in Runoff Prediction and the Value of Data: An Application of the GLUE Approach. Water Resources Research 32(7), 2161-2173.

Gan, T.Y. and Biftu, G.F. (1996) Automatic Calibration of Conceptual Rainfall-Runoff Models: Optimization Algorithms, Catchment Conditions, and Model Structure. Water Resources Research 32(12), 3513-3524.

Golberg, D.E. (1989) Genetic algorithms in search, optimization, and machine learning. Addion wesley 1989, 102.

Hadka, D., Madduri, K. and Reed, P. (2013) Scalability analysis of the asynchronous, master-slave Borg multiobjective evolutionary algorithm, pp. 425-434, IEEE.

Hadka, D. and Reed, P. (2013) Borg: An Auto-Adaptive Many-Objective Evolutionary Computing Framework. Evolutionary Computation 21(2), 231-259.

Hasalová, L., Ira, J. and Jahoda, M. (2016) Practical observations on the use of Shuffled Complex Evolution (SCE) algorithm for kinetic parameters estimation in pyrolysis modeling. Fire Safety Journal 80, 71-82.

Holland, J.H. (1975) Adaptation in Natural and Artificial Systems, Univ. of Mich. Press. Ann Arbor.

Holland, J.H. (1992) Adaptation in natural and artificial systems. 1975. Ann Arbor, MI: University of Michigan Press and.

K. Ajami, N., Gupta, H., Wagener, T. and Sorooshian, S. (2004) Calibration of a semi-distributed hydrologic model for streamflow estimation along a river system. Journal of Hydrology 298(1), 112-135.

Kan, A.R. and Timmer, G.T. (1987) Stochastic global optimization methods part I: Clustering methods. Mathematical programming 39(1), 27-56.

Kaveh, A. and Farhoudi, N. (2013) A new optimization method: Dolphin echolocation. Advances in Engineering Software 59, 53-70.

Kaveh, A. and Talatahari, S. (2010) A novel heuristic optimization method: charged system search. Acta Mechanica 213(3), 267-289.

Kennedy, J. (2010) Encyclopedia of Machine Learning. Sammut, C. and Webb, G.I. (eds), pp. 760-766, Springer US, Boston, MA.

Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P. (1983) Optimization by simmulated annealing. science 220(4598), 671-680.

Lee, K.S. and Geem, Z.W. (2005) A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. Computer Methods in Applied Mechanics and Engineering 194(36), 3902-3933.

Liang, J.J., Suganthan, P.N. and Deb, K. (2005) Novel composition test functions for numerical global optimization, pp. 68-75.

Lin, J.-Y., Cheng, C.-T. and Chau, K.-W. (2006) Using support vector machines for long-term discharge prediction. Hydrological Sciences Journal 51(4), 599-612.

Liong, S.-Y. and Atiquzzaman, M. (2004) Optimal design of water distribution network using shuffled complex evolution. Journal of The Institution of Engineers, Singapore 44(1), 93-107.

Madsen, H. (2000) Automatic calibration of a conceptual rainfall–runoff model using multiple objectives. Journal of Hydrology 235(3), 276-288.

Madsen, H. (2003) Parameter estimation in distributed hydrological catchment modelling using automatic calibration with multiple objectives. Advances in Water Resources 26(2), 205-216.

Maier, H.R., Kapelan, Z., Kasprzyk, J., Kollat, J., Matott, L.S., Cunha, M.C., Dandy, G.C., Gibbs, M.S., Keedwell, E., Marchi, A., Ostfeld, A., Savic, D., Solomatine, D.P., Vrugt, J.A., Zecchin, A.C., Minsker, B.S., Barbour, E.J., Kuczera, G., Pasha, F., Castelletti, A., Giuliani, M. and Reed, P.M. (2014) Evolutionary algorithms and other metaheuristics in water resources: Current status, research challenges and future directions. Environmental Modelling & Software 62(Supplement C), 271-299.

Mariani, V.C., Justi Luvizotto, L.G., Guerra, F.A. and dos Santos Coelho, L. (2011) A hybrid shuffled complex evolution approach based on differential evolution for unconstrained optimization. Applied Mathematics and Computation 217(12), 5822-5829.

Mirjalili, S. and Hashim, S.Z.M. (2010) A new hybrid PSOGSA algorithm for function optimization, pp. 374-377.

Mirjalili, S., Mirjalili, S.M. and Lewis, A. (2014) Grey Wolf Optimizer. Advances in Engineering Software 69, 46-61.

Mirjalili, S., Saremi, S., Mirjalili, S.M. and Coelho, L.d.S. (2016) Multi-objective grey wolf optimizer: A novel algorithm for multi-criterion optimization. Expert Systems with Applications 47(Supplement C), 106-119.

Moeini, R. and Afshar, M. (2009) Application of an ant colony optimization algorithm for optimal operation of reservoirs: a comparative study of three proposed formulations. Sci Iran Trans A Civ Eng 16(4), 273-285.

Nelder, J.A. and Mead, R. (1965) A simplex method for function minimization. The computer journal 7(4), 308-313.

Nicklow, J., Reed, P., Savic, D., Dessalegne, T., Harrell, L., Chan-Hilton, A., Karamouz, M., Minsker, B., Ostfeld, A., Singh, A. and Zechman, E. (2010) State of the Art for Genetic Algorithms and Beyond in Water Resources Planning and Management. Journal of Water Resources Planning and Management 136(4), 412-432.

Olorunda, O. and Engelbrecht, A.P. (2008) Measuring exploration/exploitation in particle swarms using swarm diversity, pp. 1128-1134.

Omran, M.G.H., Salman, A. and Engelbrecht, A.P. (2005) Computational Intelligence and Security: International Conference, CIS 2005, Xi’an, China, December 15-19, 2005, Proceedings Part I. Hao, Y., Liu, J., Wang, Y., Cheung, Y.-m., Yin, H., Jiao, L., Ma, J. and Jiao, Y.-C. (eds), pp. 192-199, Springer Berlin Heidelberg, Berlin, Heidelberg.

Passino, K.M. (2002) Biomimicry of bacterial foraging for distributed optimization and control. IEEE Control Systems 22(3), 52-67.

Price, W. (1987) Global optimization algorithms for a CAD workstation. Journal of Optimization Theory and Applications 55(1), 133-146.

Qin, A.K. and Suganthan, P.N. (2005) Self-adaptive differential evolution algorithm for numerical optimization, pp. 1785-1791 Vol. 1782.

Rashedi, E., Nezamabadi-Pour, H. and Saryazdi, S. (2009) GSA: a gravitational search algorithm. Information sciences 179(13), 2232-2248.

Reed, P.M., Hadka, D., Herman, J.D., Kasprzyk, J.R. and Kollat, J.B. (2013) Evolutionary multiobjective optimization in water resources: The past, present, and future. Advances in Water Resources 51(Supplement C), 438-456.

Sadegh M, Vrugt JA. (2014) Approximate bayesian computation using markov chain monte carlo simulation: Dream (abc). Water Resources Research. 50(8), 6767-6787.

Sadegh M., Ragno E., AghaKouchak A. Multivariate Copula Analysis Toolbox (MvCAT): Describing dependence and underlying uncertainty using a Bayesian framework. Water Resources Research. 2017 Jun 1.

Sorooshian, S., Duan, Q. and Gupta, V.K. (1993) Calibration of rainfall-runoff models: Application of global optimization to the Sacramento Soil Moisture Accounting Model. Water Resources Research 29(4), 1185-1194.

Storn, R. and Price, K. (1997) Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. Journal of Global Optimization 11(4), 341-359.

Toth, E., Brath, A. and Montanari, A. (2000) Comparison of short-term rainfall prediction models for real-time flood forecasting. Journal of Hydrology 239(1), 132-147.

Vrugt, J.A., Gupta, H.V., Bastidas, L.A., Bouten, W. and Sorooshian, S. (2003a) Effective and efficient algorithm for multiobjective optimization of hydrologic models. Water Resources Research 39(8), n/a-n/a.

Vrugt, J.A., Gupta, H.V., Bouten, W. and Sorooshian, S. (2003b) A Shuffled Complex Evolution Metropolis algorithm for optimization and uncertainty assessment of hydrologic model parameters. Water Resources Research 39(8), n/a-n/a.

Vrugt, J.A. and Robinson, B.A. (2007) Improved evolutionary optimization from genetically adaptive multimethod search. Proceedings of the National Academy of Sciences 104(3), 708-711.

Vrugt, J.A., Robinson, B.A. and Hyman, J.M. (2009) Self-Adaptive Multimethod Search for Global Optimization in Real-Parameter Spaces. IEEE Transactions on Evolutionary Computation 13(2), 243-259.

Wagener, T., Wheater, H. and Gupta, H.V. (2004) Rainfall-runoff modelling in gauged and ungauged catchments, World Scientific.

Wang, Y.-C., Yu, P.-S. and Yang, T.-C. (2010) Comparison of genetic algorithms and shuffled complex evolution approach for calibrating distributed rainfall–runoff model. Hydrological processes 24(8), 1015-1026.

Wolpert, D.H. and Macready, W.G. (1997) No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1(1), 67-82.

Woodruff, M.J., Reed, P.M. and Simpson, T.W. (2013) Many objective visual analytics: rethinking the design of complex engineered systems. Structural and Multidisciplinary Optimization 48(1), 201-219.

Xin, Y., Yong, L. and Guangming, L. (1999) Evolutionary programming made faster. IEEE Transactions on Evolutionary Computation 3(2), 82-102.

Yang, T., Asanjan, A.A., Faridzad, M., Hayatbini, N., Gao, X. and Sorooshian, S. (2017) An enhanced artificial neural network with a shuffled complex evolutionary global optimization with principal component analysis. Information sciences 418, 302-316.

Yang, T., Gao, X., Sellars, S.L. and Sorooshian, S. (2015) Improving the multi-objective evolutionary optimization algorithm for hydropower reservoir operations in the California Oroville–Thermalito complex. Environmental Modelling & Software 69, 262-279.

Yang, X.-S. (2009) Stochastic Algorithms: Foundations and Applications: 5th International Symposium, SAGA 2009, Sapporo, Japan, October 26-28, 2009. Proceedings. Watanabe, O. and Zeugmann, T. (eds), pp. 169-178, Springer Berlin Heidelberg, Berlin, Heidelberg.

Yang, X.-S. (2010) Nature Inspired Cooperative Strategies for Optimization (NICSO 2010). González, J.R., Pelta, D.A., Cruz, C., Terrazas, G. and Krasnogor, N. (eds), pp. 65-74, Springer Berlin Heidelberg, Berlin, Heidelberg.

Yapo, P.O., Gupta, H.V. and Sorooshian, S. (1996) Automatic calibration of conceptual rainfall-runoff models: sensitivity to calibration data. Journal of Hydrology 181(1), 23-48.

Yapo, P.O., Gupta, H.V. and Sorooshian, S. (1998) Multi-objective global optimization for hydrologic models. Journal of Hydrology 204(1-4), 83-97.

Appendix A. Modified Competitive Complex Evolution (MCCE)

MCCE algorithm pseudo code is as follow:

Step 0. Initialize i = 1, and get maximum number of iteration allowed, I.

Step 1. Sort individuals in order of increasing objective function value. Assign individuals a triangular probability (except for the fittest point) according to:

$$p = \frac{2(\mathrm{NPT}+1-n)}{\mathrm{NPT}(\mathrm{NPT}+1)}, \tag{A1}$$

where NPT is the number of individuals in the complex and n is the rank of the sorted individuals.

Step 2. Select d+1 individuals (d is problem dimension) from the complex including the fittest individual in the complex.

Step 3. The selected individuals are then stored in **S**, forming a simplex. Generate offspring according to following steps.

I. Sort individuals in **S** according to their objective function value. Find centroid, $\vec{c}$, of the first q individuals.

II. Reflection: Reflect the worst individual in **S**, $\vec{w}$, across the centroid to generate a new point, $\vec{r}$, according to following equation:

$$\vec{r} = 2\vec{c} - \vec{w}. \tag{A2}$$

Evaluate objective function for the new point, f_r. If $f_1 < f_r < f_d$ set offspring, $\vec{o} = \vec{r}$, and go to (VII).

III. Expansion: If $f_r < f_1$, reflect $\vec{c}$ across $\vec{r}$ and generate $\vec{e}$,

$$\vec{e} = 2\vec{r} - \vec{c}. \tag{A3}$$

Evaluate objective function for the new point, f_e. If $f_e < f_r$, set $\vec{o} = \vec{e}$ and go to (VII); otherwise, $\vec{o} = \vec{r}$ and go to (VII).

IV. Outside contraction: If $f_d \leq f_r < f_w$, calculate the outside contraction point,

$$\overrightarrow{oc} = \vec{c} + 0.5(\vec{r} - \vec{c}). \tag{A4}$$

Evaluate the outside contraction point, f_{oc}. If $f_{oc} < f_r$ set $\vec{o} = \overrightarrow{oc}$ and go to (VII); otherwise, $\vec{o} = \vec{r}$ and go to (VII).

V. Inside contraction: If $f_w < f_r$ calculate inside contraction point,

$$\overrightarrow{\iota c} = \vec{c} + 0.5(\vec{w} - \vec{c}). \tag{A5}$$

Evaluate inside contraction point, f_{ic}. If $f_{ic} < f_r$ set $\vec{o} = \overrightarrow{\iota c}$ and go to (VII); otherwise continue to (VI).

VI. Multinormal sampling: If the steps above, did not generate a better offspring, an individual will be drawn with a multinormal distribution defined by simplex and replace the worst individual in the simplex, regardless of objective function value. The multinormal sampling is as follow,

a. Calculate the covariance matrix, R, for the simplex and store diagonal of matrix in $\vec{D}$.

b. Modify $\vec{D}$ as follow

$$\overrightarrow{D_m} = 2(\vec{D} + mean(\vec{D})). \tag{A6}$$

c. Generate a new covariance matrix R', with $\overrightarrow{D_m}$ as diagonal and zeroes everywhere else.

d. Sample a point with multinormal distribution with mean of $\vec{c}$ and covariance of R' and store in $\vec{o}$.

VII. Replace the worst individual in the complex with $\vec{o}$. Let $i = i + 1$. If $i \leq I$, go to (Step 1); otherwise sort the complex and return the evolved complex.

Appendix B. Modified Frog Leaping (MFL)

Modified FL (MFL) algorithm is as follow,

Step 0. Initialize i = 1, and get maximum number of iteration allowed , I.

Step 1. Sort individuals in order of increasing objective function. Assign individuals a triangular probability using following equation:

$$p = \frac{2(\mathrm{NPT}+1-n)}{\mathrm{NPT}(\mathrm{NPT}+1)}, \tag{B1}$$

where NPT is the number of individuals in the complex and n is the rank of the sorted individuals.

Step 2. Select d+1 individuals (d is problem dimension) from the complex.

Step 3. The selected individuals are stored in **S**, forming a subcomplex. Generate offspring according to following steps.

I. Generate a new point with the worst point in **S**, $\vec{w}$ and best point $\vec{b}$ in the subcomplex, as follow,

$$\overrightarrow{n_b} = \vec{w} + (0.5 \times R + 1.5)(\vec{b} - \vec{w}), \tag{B2}$$

where R is a random number in the range of [0,1]. Evaluate objective function for the new point and get f_b. If $f_b < f_w$; set $\vec{o} = \overrightarrow{n_b}$ and go to (IV).

II. If $f_w < f_b$, generate a new point with the worst point in **S**,$\vec{w}$ and best point $\vec{b}$ in the subcomplex, as follow,

$$\overrightarrow{n_B} = \vec{w} + 0.5 \times R(\vec{b} - \vec{w}), \tag{B3}$$

Evaluate objective function for the new point and get f_B. If $f_B < f_w$ set the offspring set $\vec{o} = \overrightarrow{n_B}$ and go to (IV).

III. Censorship step: If $f_w < f_B$, randomly generate the offspring, $\vec{o}$ by sampling within the range of individuals in the subcomplex.

IV. Replace the worst individual in the complex with the offspring, $\vec{o}$. Let $i = i + 1$. If $i \leq I$, go to (Step 1); otherwise sort the complex and return the evolved complex.

Appendix C. Modified Grey Wolf Optimizer (GWO)

Modified Grey Wolf Optimizer is as follow:

Step 0. Initialize i = 1, and get maximum number of iteration allowed, I.

Step 1. Sort the individuals in the order of increasing objective function value. Assign individuals a triangular probability (except for the fittest point) using following equation:

$$p = \frac{2(\mathrm{NPT}+1-n)}{\mathrm{NPT}(\mathrm{NPT}+1)}, \tag{C1}$$

where NPT is the number of individuals in the complex and n is the rank of the sorted individuals.

Step 2. Select d+1 individuals (d is problem dimension) from the complex, with triangular probability, including the fittest point in the complex and store them in **S**.

Step 3. Select the best three points in the **S** and store them in $\vec{\alpha}$, $\vec{\beta}$ and $\vec{\gamma}$, respectively. The worst point in the **S**, is stored in $\overrightarrow{w}$

Step 4. For each of $\vec{\alpha}$, $\vec{\beta}$ and $\vec{\gamma}$, evolve individuals according to the following procedure,

I. Derive $\vec{A}$ and $\vec{C}$ as follow for $\vec{\alpha}$, $\vec{\beta}$ and $\vec{\gamma}$,

$$\vec{A} = 4 \times \vec{r}_1 - 2, \tag{C2}$$

$$\vec{C} = 2 \times \vec{r}_2. \tag{C3}$$

where $\overrightarrow{r_1}, \overrightarrow{r_2}$ are two independent random vectors, which have d dimensions and values in range of [0,1).

II. Derive $\overrightarrow{D}$, for $\vec{\alpha}$, $\vec{\beta}$ and $\vec{\gamma}$ as follow,

$$\overrightarrow{D_\alpha} = \left|\overrightarrow{C_\alpha} \times \overrightarrow{X_\alpha} - \vec{w}\right|, \overrightarrow{D_\beta} = \left|\overrightarrow{C_\beta} \times \overrightarrow{X_\beta} - \vec{w}\right|, \overrightarrow{D_\gamma} = \left|\overrightarrow{C_\gamma} \times \overrightarrow{X_\gamma} - \vec{w}\right|. \tag{C4}$$

III. Derive $\vec{Z}$, for $\vec{\alpha}$, $\vec{\beta}$ and $\vec{\gamma}$ as follow,

$$\overrightarrow{Z_\alpha} = \overrightarrow{X_\alpha} - \overrightarrow{A_\alpha}.(\overrightarrow{D_\alpha}), \overrightarrow{Z_\beta} = \overrightarrow{X_\beta} - \overrightarrow{A_\beta}.(\overrightarrow{D_\beta}), \overrightarrow{Z_\gamma} = \overrightarrow{X_\gamma} - \overrightarrow{A_\gamma}.(\overrightarrow{D_\gamma}). \tag{C5}$$

IV. Generate new point by finding the centroid of $\overrightarrow{Z_\alpha}$, $\overrightarrow{Z_\beta}$ and $\overrightarrow{Z_\gamma}$,

$$\vec{C} = \frac{\overrightarrow{Z_\alpha}+\overrightarrow{Z_\beta}+\overrightarrow{Z_\gamma}}{3}. \tag{C6}$$

V. Calculate and store objective function value for the new point, f_C. If the new point is better than the worst point among the selected points, $f_C < f_w$, set $\vec{o} = \vec{C}$, go to step 7.

Step 5. If $f_C > f_w$, go to step 4, and use a smaller range for $\vec{A}$. In this step, $\vec{A}$ is calculated as follow:

$$\vec{A} = 2 \times \vec{r}_1 - 1, \tag{C7}$$

Step 6. If the newly generated individual is worse than the worst individuals in subcomplex, generate a new point with uniform random sampling within the range of individuals in the complex. Store the new point in $\vec{o}$.

Step 7. Replace the worst individual among selected points in the complex with the offspring, $\vec{o}$. Let $i = i + 1$. If $i \le I$, go to (Step 1); otherwise sort the complex and return the evolved complex.

Appendix D. Modified Differential Evolution (DE)

Modified differential evolution algorithm is as follow:

Step 0. Initialize i = 1, and get maximum number of iteration allowed, I.

Step 1. Sort the individuals in the order of increasing objective function value. Assign individuals a triangular probability, using following equation:

$$p = \frac{2(\mathrm{NPT}+1-n)}{\mathrm{NPT}(\mathrm{NPT}+1)}, \tag{D1}$$

where NPT is the number of individuals in the complex and n is the rank of the sorted individuals.

Step 2. Select d+1 points (d is problem dimension) from the complex with the assigned probability and store them along with the fittest point in the complex in **S**.

Step 3. The selected individuals are sorted and stored in **S**, forming a subcomplex. Generate offspring according to following steps.

I. Generate a new point with the worst point in **S**, $\vec{w}$ and using the top three individuals in the subcomplex,

$$\overrightarrow{V_1} = \vec{w} + 2f(\overrightarrow{s_1} - \vec{w}) + 2f(\overrightarrow{s_2} - \overrightarrow{s_3}), \tag{D2}$$

where $\vec{w}$ is the worst point in the **S**, $\overrightarrow{s_1}$, $\overrightarrow{s_2}$, and $\overrightarrow{s_3}$ are three selected individuals. Then mutation, and crossover operator is applied to the $\vec{w}$ and $\overrightarrow{V_1}$ to generate $\overrightarrow{V_{n1}}$. The objective function value for the new point is calculated and stored in f_{n1}. If $f_{n1} < f_w$; set $\vec{o} = \overrightarrow{V_{n1}}$ and go to (V).

II. If $f_w < f_{n1}$, generate a new point with the worst point in **S**, $\vec{w}$ and using the top three points in the subcomplex as follow,

$$\overrightarrow{V_2} = \vec{w} + 0.5f(\overrightarrow{s_1} - \vec{w}) + 0.5f(\overrightarrow{s_2} - \overrightarrow{s_3}), \tag{D3}$$

After mutation, crossover operator is applied to the $\vec{w}$ and $\overrightarrow{V_2}$ to generate $\overrightarrow{V_{n2}}$. Then, the objective function for the new point is derived and stored in f_{n2}. If $f_{n2} < f_w$; set $\vec{o} = \overrightarrow{V_{n2}}$ and go to (V).

III. If $f_w < f_{n2}$, generate a new point with the worst point in **S**, $\vec{w}$ and using the top three points in the subcomplex as follow,

$$\overrightarrow{V_3} = \vec{w} + f(\overrightarrow{s_1} - \vec{w}) + f(\overrightarrow{s_2} - \overrightarrow{s_3}), \quad \text{(D4)}$$

After mutation, crossover operator is applied to the $\vec{w}$ and $\overrightarrow{V_3}$ to generate $\overrightarrow{V_{n3}}$. The objective function value is calculated and stored in f_{n3}. If $f_{n3} < f_w$; set $\vec{o} = \overrightarrow{V_{n3}}$ and go to (V).

IV. If the newly generated point is worse than the worst point in subcomplex, generate a new point from uniform random distribution within the range of points in the complex. Store the new point in $\vec{o}$.

V. Replace the worst point in the complex with the offspring, $\vec{o}$. Let $i = i + 1$. If $i \leq I$, go to (Step 1); otherwise sort the complex and return the evolved complex.

Abbreviation

AMALGAM-SO: A Multialgorithm Genetically Adaptive Method for Single Objective Optimization

CCE: Competitive Complex Evolution

DE: Differential Evolution

EA: Evolutionary Algorithm

EMP: Evolutionary Methods Performance

FL: Frog Leaping

GWO: Grey Wolf Optimizer

LHS: Latin Hypercube Sampling

MCCE: Modified Competitive Complex Evolution

MFL: Modified Frog Leaping

MGWO: Modified Grey Wolf Optimizer

MOCOM-UA: Multi-Objective Complex evolution, University of Arizona

MOSCEM: Multi-Objective Shuffled Complex Evolution Metropolis

NFL: No Free Lunch

PCA: Principal Component Analysis

PSO: Particle Swarm Optimization

SaDE: Self-adaptive DE algorithm

SCE-UA: Shuffle Complex Evolution-developed at University of Arizona

SCEM-UA: Shuffled Complex Evolution Metropolis algorithm-developed at University of Arizona

SC-SAHEL: Shuffle Complex-Self Adaptive Hybrid EvoLution

SP-UCI: Shuffled Complex strategy with Principal component analysis-developed at University of California, Irvine

URS: Uniform Random Sampling