

Boise State University

ScholarWorks

Civil Engineering Faculty Publications and
Presentations

Department of Civil Engineering

6-2018

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>

For a complete list of authors, please see article.

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

2 **Framework**

3 **Matin Rahnamay Naeini¹, Tiantian Yang^{1,2}, Mojtaba Sadegh^{1,3}, Amir Aghakouchak¹, Kuo-**
4 **lin Hsu¹, Soroosh Sorooshian¹, Qingyun Duan⁴, and Xiaohui Lei⁵**

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

7 ²Deltares USA Inc., Silver Spring, Maryland, USA

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

9 ⁴Beijing Normal University, Faculty of Geographical Sciences. Beijing, China

10 ⁵China Institute of Water Resources and Hydropower Research, Beijing, China

11

12

13

14

15

16

17

18

19

20 Corresponding author: Tiantian Yang (tiantiy@uci.edu)

21 **Abstract**

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

34 **Keywords**

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

37

38 **Software availability**

39 Name of software: SC-SAHEL

40 Developer: Matin Rahnamay Naeini

41 Contact address: rahnamam@uci.edu

42 Program language: MATLAB

43 Year first available: 2018

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

45 Software requirements: MATLAB 9.0

46 **1 Introduction**

47 Meta-Heuristic optimization algorithms have gained a great deal of attention in science and
48 engineering (Blum and Roli 2003, Boussaïd et al. 2013, Lee and Geem 2005, Maier et al. 2014,
49 Nicklow et al. 2010, Reed et al. 2013). Simplicity and flexibility of these algorithms, along with
50 their robustness make them attractive tools for solving optimization problems (Coello et al. 2007,
51 Lee and Geem 2005). Many of the meta-heuristic algorithms are inspired by a physical
52 phenomenon, such as animals social and foraging behavior and natural selection. For example,
53 Simulated Annealing (Kirkpatrick et al. 1983), Big Bang-Big Crunch (Erol and Eksin 2006),
54 Gravitational Search Algorithm (Rashedi et al. 2009), Charged System Search (Kaveh and
55 Talatahari 2010) are inspired by various physical phenomena. Ant Colony Optimization (Dorigo
56 et al. 1996), Particle Swarm Optimization (Kennedy 2010), Bat-inspired Algorithm (Yang 2010),
57 Firefly Algorithm (Yang 2009), Dolphin Echolocation (Kaveh and Farhoudi 2013), Grey Wolf
58 Optimizer (Mirjalili et al. 2014), Bacterial Foraging (Passino 2002), Genetic Algorithm (Golberg
59 1989, Holland 1992), and Differential Evolution (Storn and Price 1997) are examples of algorithms
60 inspired by animal's social and foraging behavior, and the natural selection mechanism of
61 Darwin's Evolution Theorem. According to the No-Free-Lunch (NFL) (Wolpert and Macready
62 1997) theorem, none of these algorithms are consistently superior to others over a variety of
63 problems, although some of them may outperform on a certain type of optimization problem.

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

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

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

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

94 2011, Gan and Biftu 1996, Wagener et al. 2004, Wang et al. 2010). The SCE-UA algorithm has
95 been widely used in water resources management (Barati et al. 2014, Eckhardt and Arnold 2001,
96 K. Ajami et al. 2004, Lin et al. 2006, Liong and Atiquzzaman 2004, Madsen 2000, Sorooshian et
97 al. 1993, Toth et al. 2000, Yang et al. 2015, Yapo et al. 1996), as well as other fields of study, such
98 as pyrolysis modeling (Ding et al. 2016, Hasalová et al. 2016) and Artificial Intelligence (Yang et
99 al. 2017).

100 Application of the SCE-UA is not limited to solving single objective optimization
101 problems. The Multi-Objective Complex evolution, University of Arizona (MOCOM-UA), is an
102 extension of the SCE-UA for solving multi-objective problems (Boyle et al. 2000, Yapo et al.
103 1998). Besides, the SCE-UA architecture has been used to develop Markov Chain Monte Carlo
104 (MCMC) sampling, named Shuffled Complex Evolution Metropolis algorithm (SCEM-UA) and
105 the Multi-Objective Shuffled Complex Evolution Metropolis (MOSCEM) to infer posterior
106 parameter distribution of hydrologic models (Vrugt et al. 2003a, Vrugt et al. 2003b). The
107 Metropolis scheme is used as the search kernel in the SCEM-UA and MOSCEM-UA (Chu et al.
108 2010, Vrugt et al. 2003a, Vrugt et al. 2003b). There is also an enhanced version of SCE-UA, which
109 is developed by Chu et al. (2011) entitled the Shuffled Complex strategy with Principle Component
110 Analysis, developed at the University of California, Irvine (SP-UCI). Chu et al. (2011) found that
111 the SCE-UA algorithm may not converge to the best solution on high-dimensional problems due
112 to “population degeneration” phenomenon. The “population degeneration” refers to the situation
113 when the search particles span a lower dimension space than the original search space (Chu et al.
114 2010), which causes the search algorithm to fail in finding the global optimum. To address this
115 issue, the SP-UCI algorithm employs Principle Component Analysis (PCA) in order to find and
116 restore the missing dimensions during the course of search (Chu et al. 2011).

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

125 The Competitive Complex Evolution (CCE) and Modified Competitive Complex
126 Evolution (MCCE) are the search cores of the SCE-UA and SP-UCI algorithm, respectively. The
127 CCE and MCCE evolutionary processes are developed based on Nelder-Mead (Nelder and Mead
128 1965) method with some modification. The evolution process in the SCE-UA is not limited to
129 these algorithms. In fact, several studies have incorporated different EAs into the structure of the
130 SCE-UA algorithm. For example, the Frog Leaping (FL) is developed by adapting Particle Swarm
131 Optimization (PSO) algorithm to the SCE-UA structure for solving discrete problems (Eusuff et
132 al. 2006, Eusuff and Lansey 2003). Mariani et al. (2011) proposed an SCE-UA algorithm which
133 employs DE for evolving the complexes. These studies revealed the flexibility of the SCE-UA in
134 combination with other types of EAs; however, the potential of combining different algorithms
135 into a hybrid shuffled complex scheme has not been investigated.

136 The unique structure of the SCE-UA algorithm along with the flexibility of the algorithm
137 for using different EAs, motivated us to use the SCE-UA as the cornerstone of the SC-SAHEL
138 framework. The SC-SAHEL algorithm employs multiple EAs for evolving the population in a
139 similar structure as that of the SCE-UA, with the goal of selecting the most suitable search
140 algorithm at each optimization step. On the one hand, some EAs are more capable of visiting the

141 new regions of the search space and exploring the problem space, and hence are particularly
142 suitable at the beginning of the optimization (Olorunda and Engelbrecht 2008). On the other hand,
143 some EAs are more capable of searching within the visited regions of the search space, and hence
144 boosting the convergence process after finding the region of interest (Mirjalili and Hashim 2010).
145 Balancing between these two steps, which are referred to as exploration and exploitation (Moeini
146 and Afshar 2009), is a challenging task in stochastic optimization methods (Črepinšek et al. 2013).
147 The SC-SAHHEL algorithm maintains a balance between exploration and exploitation phases by
148 evaluating the performance of participating EAs at each optimization step. EAs contribute to the
149 population evolution according to their performance in previous steps. The algorithms'
150 performance is evaluated by comparing the evolved complexes before and after evolution. In this
151 process, the most suitable algorithm for the problem space become the dominant search core.

152 In this study, four different EAs are used as search cores in the proposed SC-SAHHEL
153 framework, including Modified Competitive Complex Evolution (MCCE) used in the SP-UCI
154 algorithm, Modified Frog Leaping (MFL), Modified Grey Wolf Optimizer (MGWO), and
155 Differential Evolution (DE). To better illustrate the performance of the hybrid SC-SAHHEL
156 algorithm, the framework is benchmarked over 29 test functions and compared to SC-SAHHEL with
157 single EA. Among the 29 employed test functions, there are 23 classic test functions (Xin et al.
158 1999) and 6 composite test functions (Liang et al. 2005), which are commonly used as benchmarks
159 in comparing optimization algorithms.

160 Furthermore, the SC-SAHHEL framework is tested for a conceptual hydropower model,
161 which is built for the Folsom reservoir located in the northern California, USA. The objective is
162 to maximize the hydropower generation, by finding the optimum discharge from the reservoir. The
163 study period covers run-off season in California from April to June, in which reservoirs have the
164 highest annual storage volume (Field and Lund 2006). Using the proposed framework, we

165 compared different EAs' capability of finding a near-optimum solution for dry, wet, and below-
166 normal scenarios. The results support that the proposed algorithm is not only competitive in terms
167 of increasing power generation, but also is able to reveal the advantages and disadvantages of
168 participating EAs.

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

175

176 **2 Methodology**

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

186

187 2.1 The SC-SAHEL framework

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

$$204 \text{ EMP} = \frac{\text{mean}(F) - \text{mean}(F_N)}{\text{mean}(F)}, \quad (1)$$

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

207 The EMP metric measures change in the mean objective function value of individuals in
208 each complex in comparison to their previous state. A higher EMP value indicates a larger
209 reduction in the mean objective function value obtained by the individuals in the complex. The

210 performance of each evolutionary algorithm is then evaluated based on the mean value of EMP
211 calculated for each evolved complex. The algorithms are then ranked according to the EMP values.
212 Ranks are in turn used to assign number of complexes to each evolutionary method for the next
213 iteration. The highest ranked algorithm will be assigned an additional complex to evolve in the
214 next shuffling step, while, the lowest ranked evolutionary algorithm will lose one complex for the
215 next step. For instance, if all the EAs have 2 complexes to evolve (2-2-2-2 case), the number of
216 complexes assigned to each EA can be updated to 3-2-2-1. In other words, this logic is an “award
217 and punishment” process, in which the algorithm with best performances will be “awarded” with
218 an additional complex to evolve in the next iteration, while the worst-performing algorithm will
219 be “punished” by losing one complex.

220 It is worth mentioning that as some of the algorithms may have poor performance in the
221 exploration phase, they might lose all their complexes during the adaptation process. This might
222 be troublesome as these algorithms may be superior in the exploitation phase. If use of such
223 algorithms are terminated in the exploration phase, they cannot be selected during the convergence
224 steps. Hence, EAs termination is avoided to fully utilize the potential of EAs in all the optimization
225 steps and balance the exploration and exploitation phases. The minimum number of complexes
226 assigned to each evolutionary method is restricted to at least 1 complex in this case. If the lowest
227 ranked EA has only 1 complex to evolve, it won't lose its last complex. If an algorithm outperforms
228 others throughout the evolution of complexes, the number of complexes assigned to the superior
229 EA will be equal to the total number of complexes minus the number of EAs plus one. In this case,
230 all other algorithms are evolving one complex only. As all algorithms are evolving at least one
231 complex, they have the chance to outperform other EAs and gain more complexes during the
232 optimization process, and to potentially become the dominant search method as the search

233 continues toward exploitation phase. Figure 1 briefly shows the flowchart of the SC-SAHEL
234 algorithm, pseudo code of which is as follows:

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

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

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

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

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

246 *Step 5.* Evolve each complex using the corresponding EA.

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

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

252 *Step 8.* Shuffle complexes and form a new population.

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

254 SC-SAHEL allows for different settings that can influence the performance of the
255 algorithm. Careful consideration should be devoted to the selection of these settings, including
256 number of complexes, number of individuals within each complex, number of evolution steps

257 before each shuffling, and stopping criteria thresholds. Some of these settings are adopted from
258 the suggested settings for the SCE-UA. For instance, the number of points within each complex is
259 set to $2d + 1$, where d is dimension of the problem. However, some of the suggested settings
260 cannot be applied to the SC-SAHEL framework due to use of different EAs. These settings can be
261 changed according to the complexity of the problem and the EAs used within the framework. For
262 instance, the number of complexes, the number of points within each complex, and the number of
263 evolution steps before each shuffling are problem dependent.

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

270 2.2 Evolutionary algorithms used in the SC-SAHEL

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

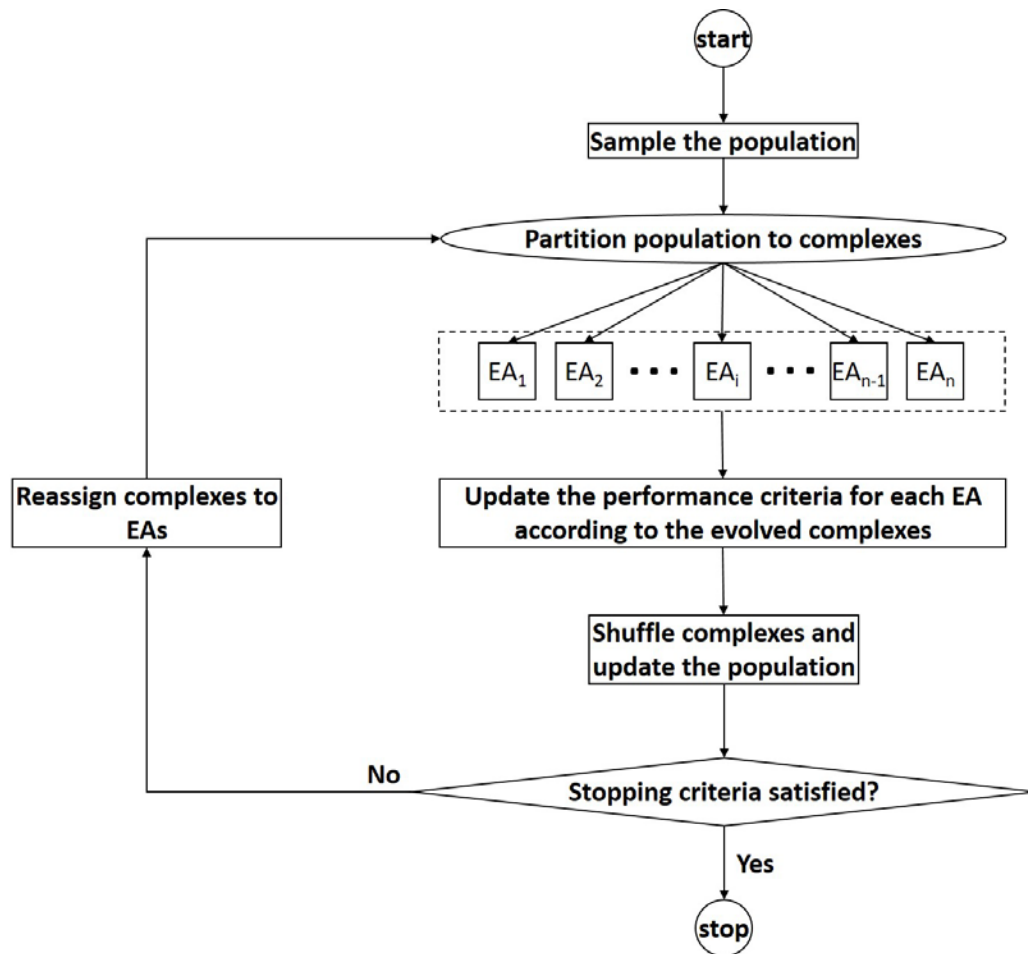


Figure 1. The SC-SAHEL framework flowchart

275

276

277 2.2.1 Modified Competitive Complex Evolution (MCCE)

278

279

280

281

282

283

284

285

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

286 and comparison, SC-SAHEL with the MCCE algorithm as search core is referred as SP-UCI,
287 hereafter.

288

289 2.2.2 Modified Frog Leaping (MFL)

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

310 feasible parameters range. Resampling within the whole parameter space can decrease the
311 convergence speed of the FL. Hence, the resampling process is carried out only within the range
312 of points in the complex. Hereafter, the SC-SAHEL with MFL algorithm as the only search core
313 is referred as SC-MFL.

314

315 2.2.3 Modified Grey Wolf Optimizer (MGWO)

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

333

334 2.2.4 Differential Evolution (DE)

335 The DE algorithm is a powerful but simple heuristic population-based optimization
336 algorithm (Omran et al. 2005, Sadegh and Vrugt 2014) proposed by Storn and Price (1997). In
337 2011, Mariani et al. (2011) integrated the DE algorithm into SCE-UA framework and showed that
338 the new framework is able to provide more robust solutions for some optimization problems in
339 comparison to the SCE-UA. Similar to the work by Mariani et al. (2011), we use a slightly
340 modified DE algorithm based on the concepts from Omran et al. (2005), in order to integrate the
341 DE algorithm into the SC-SAHEL framework. As the DE algorithm has slower performance in
342 comparison to other EAs used here, we have added multiple steps to the DE. Here, the DE
343 algorithm uses three different mutation rates in three attempts. In the first attempt, the algorithm
344 uses a larger mutation rate. This helps exploring the search space with larger jump rates. In the
345 second attempt, the algorithm reduces the mutation rate to a quarter of the first attempt. This will
346 enhance the exploitation capability of the EA. If none of these mutation rates could generate a
347 better offspring than the parents, in the next attempt the mutation rate is set to half of the first
348 attempt. Lastly, if none of these attempts generate a better offspring in comparison to the parents,
349 a new point is randomly selected within the range of individuals in the complex. The pseudo code
350 of the modified DE algorithm is detailed in Appendix D. The SC-SAHEL algorithm is referred to
351 as SC-DE, when the DE algorithm is used as the only search algorithm.

352 **3 Conceptual test functions and results**

353 3.1 Test functions

354 The SC-SAHEL framework is benchmarked over 29 mathematical test functions using
355 single-method and multi-method search mechanisms. This includes 23 classic test functions
356 obtained from Xin et al. (1999). The name and formulation of these functions along with their

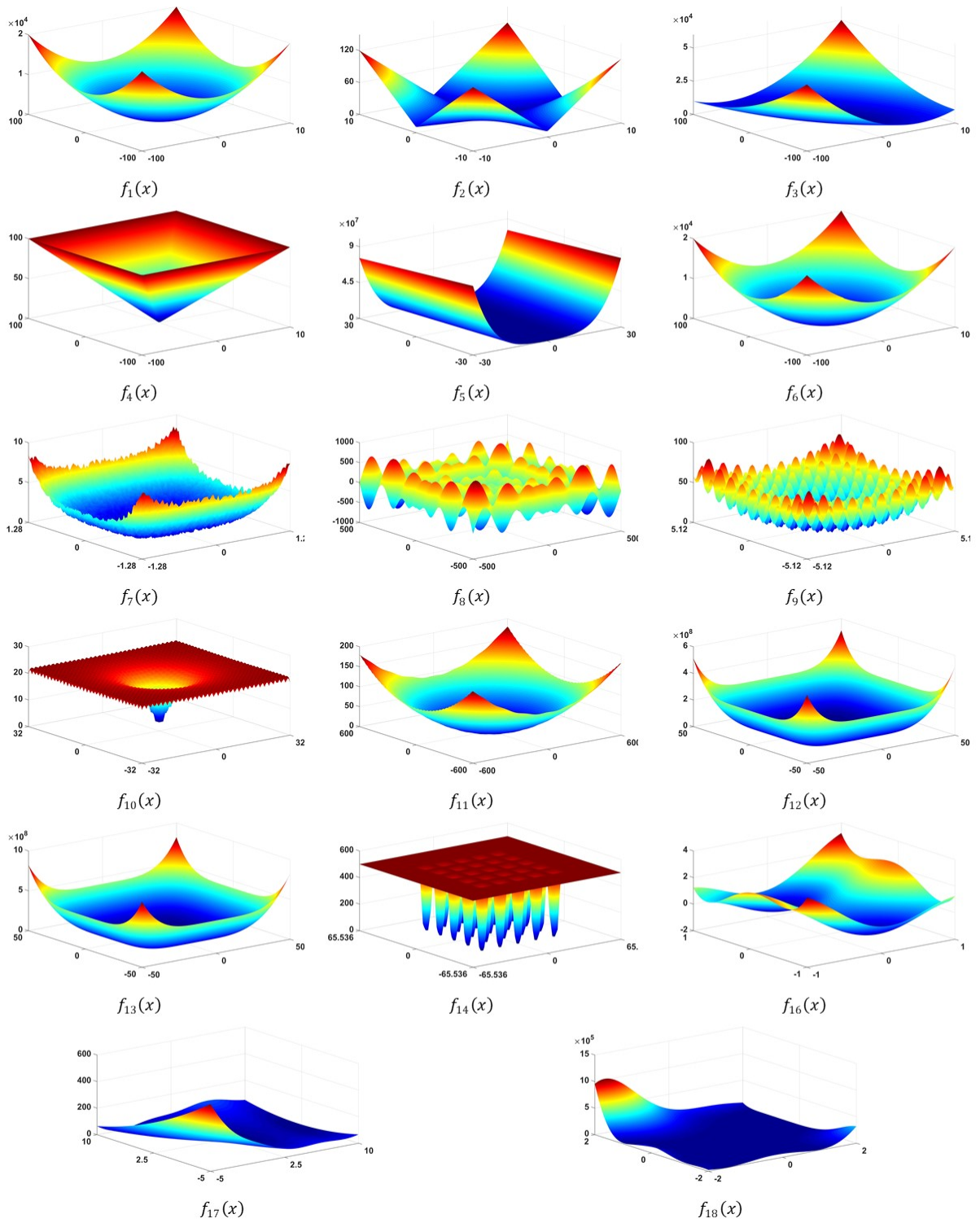
357 dimensionality and range of parameters are listed in Table 1. We selected these test functions as
358 they are standard and popular benchmarks for evaluating new optimization algorithms (Mirjalili
359 et al. 2014). The remaining 6 are composite test functions, cf_{1-6} , (Liang et al. 2005), which
360 represent complex optimization problems. Details of the composite test functions can be found in
361 the work of Liang et al. (2005) and Mirjalili et al. (2014). Classic test functions have dimensions
362 in the range of 2 to 30, and all the composite test functions are 10 dimensional. Figures 2 and 3
363 show response surface of these test functions in 2-dimension form. The SC-SAHEL settings used
364 for optimizing these test functions are listed in Table 2 for each test function. Number of points in
365 each complex and number of evolution steps for each complex are set to $2d+1$ and $\max(d+1,10)$,
366 respectively, where d is the dimension of the problem. The number of evolution steps is set to
367 $\max(d+1,10)$, to guarantee that EAs evolve the complexes for enough number of steps, before
368 evaluating the EAs. In the high-dimension problems, the maximum number of function evaluation
369 should be selected with careful consideration.

370 Several experiments were conducted to find an optimal set of parameters for the SC-
371 SAHEL setting. These experiments revealed that a low number of evolutionary steps before
372 shuffling the complexes, may not show the potential of the EAs. On the other hand, using a large
373 value for the number of evolution steps may shrink the complex to a small space, which cannot
374 span the whole search space (Duan et al. 1994). Maximum number of function evaluation is
375 determined according to the complexity of the problem and is different for each of the test cases.
376 In addition to the maximum number of function evaluation, the range of the parameters in the
377 population and the improvement in the objective function values are used as convergence criteria.
378 The optimization run is terminated if the population range is smaller than $10^{-7}\%$ of the feasible
379 range or the improvement in (objective) function value is smaller than 0.1% of the mean (objective)
380 function value in the last 50 shuffling steps. The LHS mechanism is used as the sampling algorithm

381 of SC-SAHEL for generating the initial population. The framework provides multiple settings for
382 boundary handling, which can be selected by the user. SC-SAHEL uses reflection as the default
383 boundary handling method. Other initial sampling and boundary handling methods are also
384 implemented in the SC-SAHEL framework. Sensitivity of the initial sampling and boundary
385 handling on the performance of the SC-SAHEL algorithm is not studied in this paper. The
386 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 x_i + \prod_{i=1}^n x_i $	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	$f(x) = \max_i \{ x_i , 1 \leq i \leq n\}$	30	[-100,100]	0
$f_5(x)$	Generalized Rosenbrock's Function	$f(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	[-30,30]	0
$f_6(x)$	Step Function	$f(x) = \sum_{i=1}^n (x_i + 0.5)^2$	30	[-100,100]	0
$f_7(x)$	Quartic Function	$f(x) = \sum_{i=1}^n ix_i^4 + \text{random}[0,1]$	30	[-1.28,1.28]	0
$f_8(x)$	Generalized Schwefel's Problem 2.26	$f(x) = \sum_{i=1}^n -x_i \sin(\sqrt{ x_i })$	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	$f(x) = \frac{1}{500 + \sum_{j=1}^{25} \frac{1}{\sum_{i=1}^2 (x_i - a_{ij})^6}}$	2	[-65.536,65.536]	1
$f_{15}(x)$	Kowalik's Function	$f(x) = \sum_{i=1}^{11} \left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4} \right]^2$	4	[-5,5]	0.0003075
$f_{16}(x)$	Six-Hump Camel-Back Function	$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	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	$f(x) = -\sum_{i=1}^4 c_i \exp\left[-\sum_{j=1}^4 a_{ij}(x_j - p_{ij})^2\right]$	4	[0,1]	-3.86
$f_{20}(x)$	Hartman's Family	$f(x) = -\sum_{i=1}^4 c_i \exp\left[-\sum_{j=1}^6 a_{ij}(x_j - p_{ij})^2\right]$	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

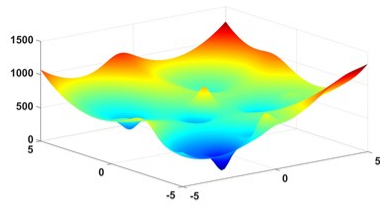


389

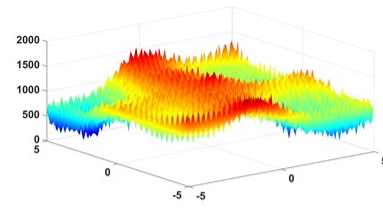
390

Figure 2. Classic test functions in 2-dimension form

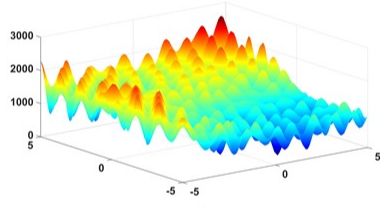
391



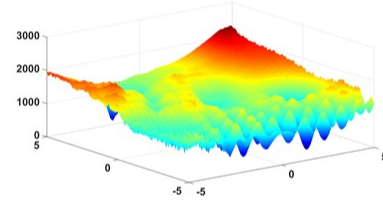
cf_1



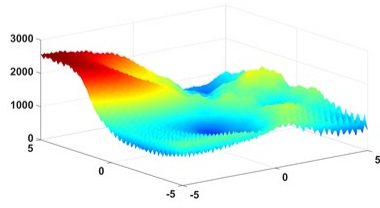
cf_2



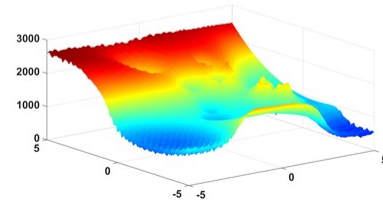
cf_3



cf_4



cf_5



cf_6

392

393

394

Figure 3. Composite test functions in 2-dimension form

Table 2. List of the settings for the SC-SAHHEL 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

397 3.2 Results and Discussion

398 Table 3 illustrates the statistics of the final function values at 30 independent runs on 29
399 test functions using the hybrid SC-SAHHEL and individual EAs, with the goal to minimize the
400 function values. The best mean function value obtained for each test function is expressed in bold
401 in Table 3. Results show that the hybrid SC-SAHHEL achieved the lowest function values in 15 out
402 of 29 test functions, compared to the mean function values achieved by all individual algorithms.
403 It is noteworthy that in 20 out of 29 test functions, the hybrid SC-SAHHEL was among the top two
404 optimization methods in finding the minimum function value. A two-sample t-test (with 5%
405 significance level) also showed that the result generated with the SC-SAHHEL algorithm is

406 generally similar to the best performing algorithms. Comparing among single-method algorithms,
 407 in general, the statistics obtained by SP-UCI are superior to other participating EAs. In 12 out of
 408 29 test functions, the SP-UCI algorithm achieved the lowest function value. SC-MFL, SC-MGWO,
 409 and SC-DE were superior to other algorithms in 10, 11, and 6 out of 29 test functions, respectively.
 410 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
 411 achieved same function values on average in most cases. In these cases, according to the statistics
 412 shown in Table 3, the SP-UCI and SC-SAHEL algorithms offer lower standard deviation values
 413 and show more consistent results as compared to other EAs. The low standard deviation values
 414 obtained by SP-UCI and SC-SAHEL indicate the robustness and consistency of these two
 415 algorithms in comparison to other algorithms.

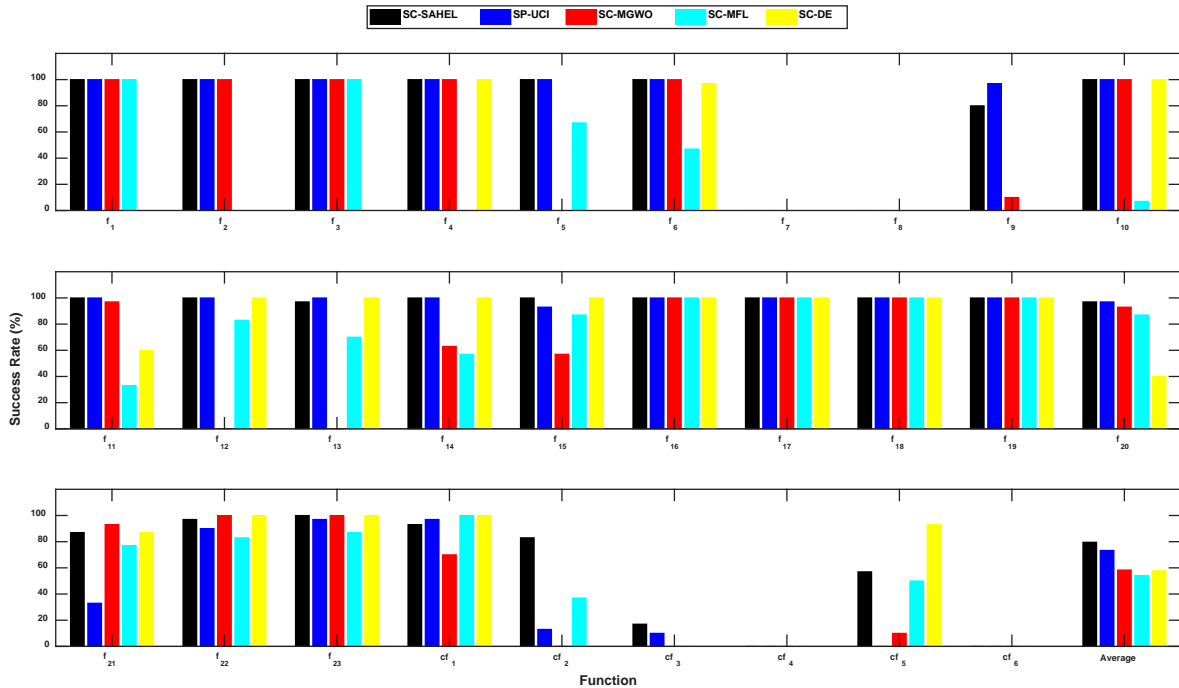
416 **Table 3. The mean and Standard deviation (Std) of objective function values for 30 independent runs on 29 test functions**
 417 **using the SC-SAHEL algorithm with single-method and multi-method search mechanism.**

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

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

442 further investigations based on the problem space and EAs. However, the EMP metric seems to be
443 a suitable metric for a wide range of problems.

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



461
462
463 **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**

464 In some situations, the poor performing EAs may mislead other EAs and cause early (and
465 premature) convergence. For instance, on the test function cf_5 , the hybrid algorithm achieved 57%
466 success rate, which is still better success rate than SP-UCI, SC-MFL and SC-MGWO, which are
467 0%, 10%, and 50%, respectively. On this test function (cf_5), the performance of the hybrid SC-
468 SAHEL is less affected by the most successful algorithm (DE). This may be due to the low
469 evolution speed of the DE algorithm, as the SC-SAHEL algorithm maintains both convergence
470 speed and efficiency during the entire search. The hybrid SC-SAHEL presents promising
471 performance on the test functions cf_2 and cf_3 . On test functions cf_2 and cf_3 , the success rate of
472 hybrid SC-SAHEL is significantly higher than other EAs, most of which have 0% success rates.
473 For test function cf_2 , the SC-DE algorithm achieved the lowest objective function value and the
474 highest success rate (37%) among single-method algorithms. However, when EAs are combined
475 in the hybrid form, the objective function value and the success rate are significantly improved.

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

478 **Table 4. The mean and Standard deviation (Std) of the number of function evaluation for 30 independent runs for 29 test**
 479 **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	<u>26877.93</u>	609.7676	100199.9	129.6894	33012.97	284.4416	100325.5	144.6016
f_2	39298.4	917.7627	<u>29333.33</u>	674.156	100193.7	126.3259	35876.77	344.1018	100307.9	168.2884
f_3	91746.23	2288.806	<u>73474.5</u>	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	<u>37987.4</u>	1170.002	227316.4	5488.107
f_5	335364.2	8102.236	401124.8	14599.33	439093.2	47901.23	<u>118900.7</u>	38671.69	500310.9	163.5498
f_6	40293.63	377.0177	<u>32537.93</u>	151.8836	66102.9	3708.332	43063.63	1463.074	90205.23	5773.92
f_7	<u>69779.5</u>	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	<u>45254.8</u>	15104.68	62555.83	21579.07
f_9	59710.57	12460.93	<u>33949.6</u>	996.5881	100705.8	22607.84	85055.73	20771.06	90930.3	34180.61
f_{10}	33765.77	887.3708	<u>27116.33</u>	379.9873	77520.6	15528.44	33181.03	416.0297	165489.4	4726.909
f_{11}	35504.9	629.8192	<u>30623.53</u>	860.8274	117357.6	13250.36	38652.4	19330.05	155148.6	16730.48
f_{12}	55908.07	4735.601	<u>39264.23</u>	3125.88	141722.1	31245.81	88234.23	28948.91	181820.6	5132.966
f_{13}	54148.7	3949.577	<u>32262.23</u>	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	<u>4530.2</u>	322.0474
f_{15}	9059.8	551.1741	<u>6517.167</u>	2358.518	8144.667	1151.183	66441.3	35455.39	18813.63	659.7369
f_{16}	3700.133	1115.033	<u>2746.3</u>	747.3937	3491.933	574.5392	8549.4	1494.076	3490.733	556.9577
f_{17}	3665.533	601.3615	<u>2910.633</u>	624.4692	3552.267	538.6385	11453.13	2592.687	5115.367	2082.727
f_{18}	2837.933	308.7723	<u>2000.633</u>	151.8512	2899.567	299.5645	8405.933	1320.571	2833.5	111.3877
f_{19}	4225.733	424.8389	<u>2852.2</u>	95.83045	4233.4	238.4404	13183.77	519.0798	4983.9	179.5704
f_{20}	8915.833	1069.182	<u>5645.567</u>	268.4028	8858.967	300.0987	17143.33	1316.025	12691.67	1818.526
f_{21}	7455.033	1741.525	<u>7377.533</u>	3260.208	7471.4	1554.087	18771.33	2996.925	10755.57	1573.865
f_{22}	6370.5	869.9209	<u>4512.433</u>	1290.258	7541.433	1582.216	17466.23	834.9485	8728.7	927.622
f_{23}	6200.133	614.6406	<u>4084.233</u>	464.9113	6823.7	327.7709	17351.87	861.8541	8398.067	599.0103
cf_1	15049.43	875.8969	<u>10293.43</u>	233.9694	21663.47	921.1805	74089.17	17803.2	28321.6	678.283
cf_2	16527.63	1432.21	<u>10586.8</u>	464.8359	20285.83	2346.093	36617.1	13118.85	30686.4	9354.096
cf_3	25991.03	8041.928	<u>16021.5</u>	3833.203	23801.2	3604.495	19323.13	6052.813	29496.9	9113.814
cf_4	22873.87	4414.168	<u>16510.13</u>	4052.642	21121.93	2417.582	23841.93	8026.638	35134.33	14468.31
cf_5	17044.53	1350.845	<u>13512.2</u>	746.6731	21400.57	1759.215	53551.43	23577.95	39200.77	4125.908
cf_6	13779.33	2279.744	<u>10518.1</u>	2977.194	14967.5	2820.062	22265.8	15340.72	27734.83	4606.317

480
 481 In Table 4, we present the mean and standard deviation of the number of function
 482 evaluation, which indicates the speed of each algorithm. As one of the stopping criteria in SC-
 483 SAHEL framework is the maximum number of function evaluation, some algorithms may
 484 terminate before they show their full potential. For instance, the SC-DE and the SC-MFL, usually
 485 reach the maximum number of function evaluations, while other algorithms satisfy other
 486 convergence criteria in much less number of function evaluations. In this case, the objective
 487 function value doesn't represent the potential of the slow algorithms. To give a better insight into

488 this matter, the mean and standard deviation (Std) of the number of function evaluations are
489 compared in Table 4. The goal is to compare the speed of the individual EAs and the hybrid
490 optimization algorithm. According to Table 4, in most of the test cases, the SP-UCI algorithm has
491 the least number of function evaluations, regardless of the objective function value achieved by
492 the EAs.

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

503 Figures 5, 6, and 7 compare the average number of complexes assigned to each EA for the
504 29 employed test functions during the course of the search. The variation of the number of
505 complexes assigned to each EA indicates the dominance of each EA during the course of the
506 search. Hence, the performance of EAs at each optimization step can be monitored. In many test
507 cases, MCCE (SP-UCI) algorithm has a relatively higher number of complexes than other EAs
508 during the search. This shows that MCCE is a dominant search algorithm on most of the test
509 functions. However, in some other cases, MCCE is only dominant in a certain period of the search,
510 while other EAs have demonstrated better efficiency during the entire search. For example, on test
511 functions f_7 and f_{20} , MCCE algorithm appears to be dominant only during the beginning of the

512 search. In the test function f_7 , the exploration process starts with the dominance of the MCCE and
513 shifts between MGWO and MFL after the first 20 shuffling steps. In some of the test functions,
514 such as f_7 , a more random fluctuation is observed in the number of complexes assigned to each
515 EA. The reason for this behavior is that EAs have very close competition in these shuffling steps.
516 Due to the noisy response surface of the test function f_7 , most of the EAs cannot significantly
517 improve the (objective) function values during the exploitation phase. On test functions f_8 and f_{18} ,
518 the MFL and DE algorithms are the dominant, respectively, during the beginning of the run, while
519 MCCE algorithm becomes dominant only when the algorithm is in exploitation phase. Lastly, on
520 test functions f_9 , f_{22} , cf_1 , and cf_4 , the variations of the number of complexes and the precedence
521 of different EAs as the most dominant search algorithm are observed.

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

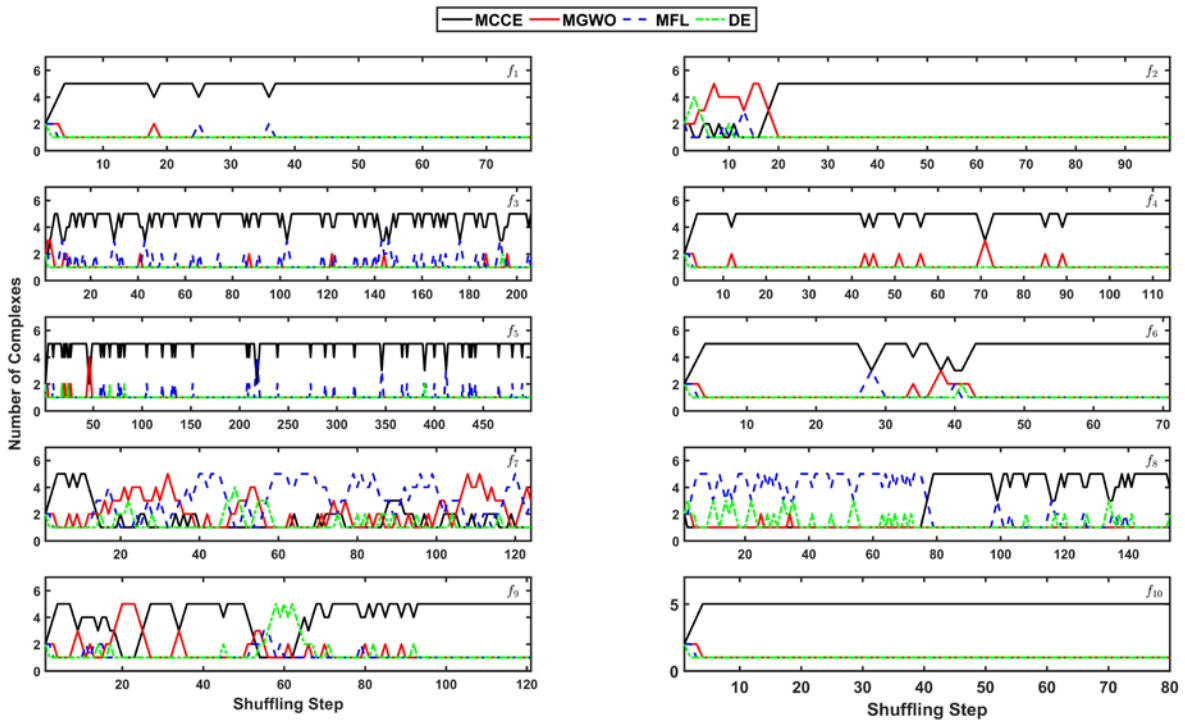


Figure 5. Number of complexes assigned to EAs during the entire optimization process on test function f_1 - f_{10}

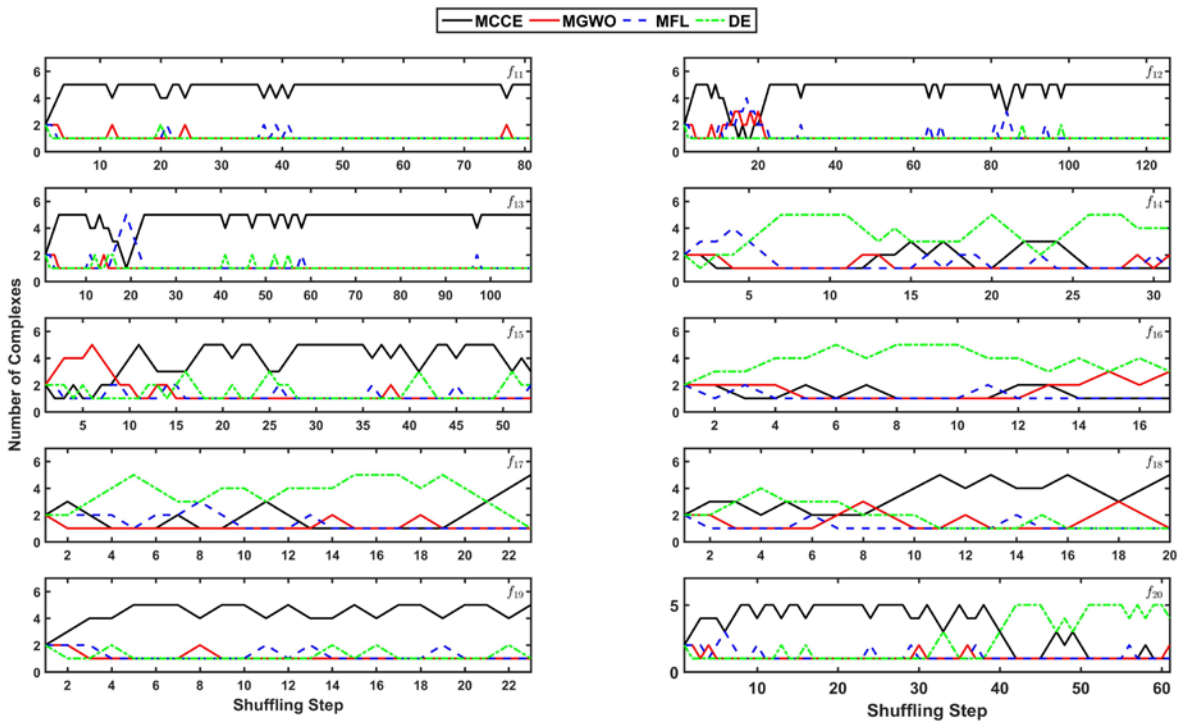
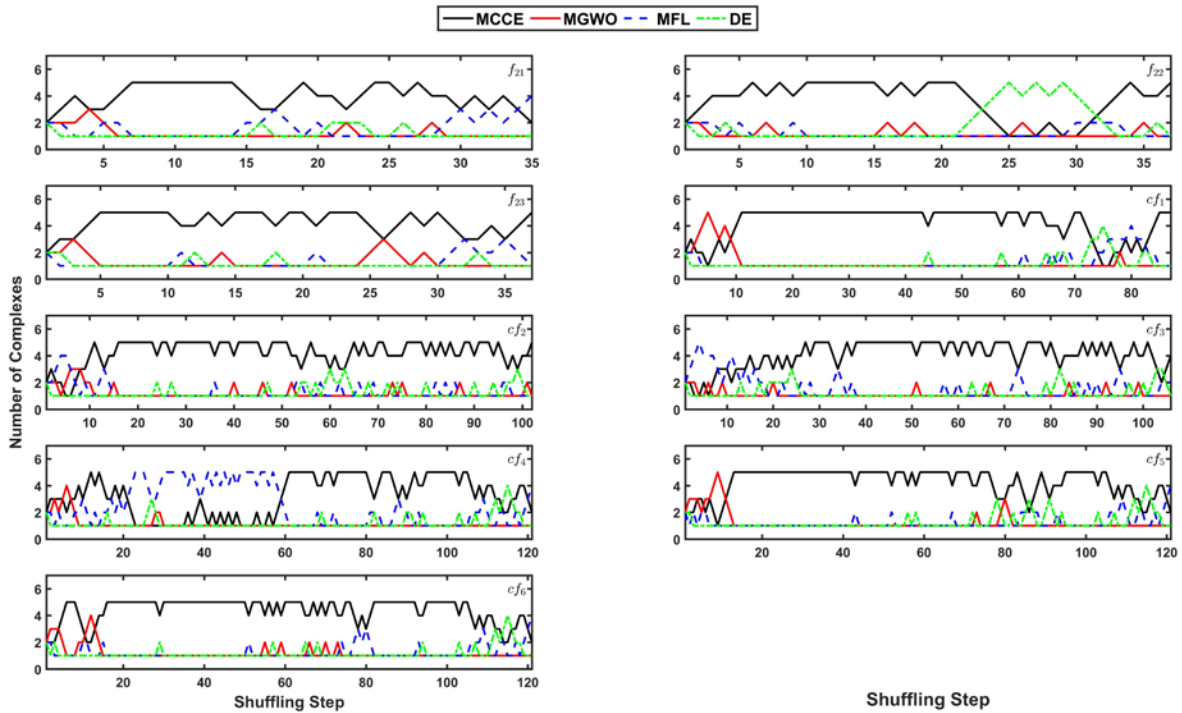


Figure 6. Number of complexes assigned to EAs during the entire optimization process on test function f_{11} - f_{20}



535

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

538

539

540

541

542

543

544

545

546

547

548

549

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.

550 **4 Example applications and results**

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

555 4.1 Reservoir Model

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

$$563 S_t = S_{t-1} + I_t - R_t \pm M_t, \quad (2)$$

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

$$567 OF = \sum_{t=1}^N 1 - \frac{P_t}{P_c}, \quad (3)$$

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

$$570 P_t = \eta \rho g Q_t H_t, \quad (4)$$

571 where η signifies turbine efficiency, ρ is water density (Kg/m³), g is gravity (9.81 m/s²) and Q_t is
572 discharge (m³/s) at time step t . H_t is hydraulic head (m) at time step t , which is defined as,

573 $H_t = h_f - h_{tw}$, (5)

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

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

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

596

597 4.2 Study basin

598 Folsom reservoir is located on the American river, in northern California and near
599 Sacramento, California. Folsom dam was built by the US Army Corps of Engineers during 1948
600 to 1956, and is a multi-purpose facility. The main functions of the facility are flood control, water
601 supply for irrigation, hydropower generation, maintaining environmental flow, water quality
602 purposes, and providing recreational area. The reservoir has a capacity of 1,203,878,290 m³ and
603 the power plant has a total capacity of 198.7 MW. Three different periods are considered here. The
604 first study period is April 1st, 2010 to June 30th, 2010. The year 2010 is categorized as below-
605 normal period according to California Department of Water Resources. The same period is
606 selected in 2011 and 2015, as former is categorized by California Department of Water Resources
607 as wet, and latter is classified as critical dry year. The input and output from the reservoir are
608 obtained from California Data Exchange Center (CDEC). Note that demand is not included in the
609 model because the demand data was not available from a public data source.

610

611 4.3 Results and Discussion

612 The boxplot of the objective function values is shown in Figure 8 for the Folsom reservoir
613 during the runoff season in 2015, 2010, and 2011, which are dry, below-normal, and wet years,
614 respectively. The presented results are based on 30 independent optimization runs; however,
615 infeasible objective function values are removed. The feasibility of the solution is evaluated
616 according to the objective function values. Due to the large values returned by the penalty function
617 considered for infeasible solutions, such solutions can be distinguished from the feasible solutions.
618 For wet year (2011) case, SC-MGWO, and SC-DE didn't find a feasible solution in 2, and 4 runs
619 out of 30 independent runs, respectively. The hybrid SC-SAHHEL found feasible solutions in all

620 the cases; however, some of these solutions are not global optima. On average, the hybrid SC-
621 SAHEL algorithm is able to achieve the lowest objective function value as compared to other
622 algorithms during dry and below-normal period. During dry and below-normal periods, SC-
623 SAHEL, SP-UCI, and SC-DE show similar performance. In the wet period, the SP-UCI algorithm
624 achieved the lowest objective function value. The SC-SAHEL algorithm ranked second,
625 comparing the mean objective function values. In this period, the results achieved by the SC-DE
626 is also comparable to SC-SAHEL and SP-UCI. The results show that overall, the hybrid SC-
627 SAHEL algorithm has similar or superior performance in comparison to the single-method
628 algorithms. Also, the results achieved by SC-SAHEL and SP-UCI algorithms has less variability
629 in comparison to other algorithms, which show the robustness of these algorithms. The worst
630 performing algorithm is the SC-MGWO, which achieved the least mean objective function value
631 in all the study periods.

632 In Figure 9, boxplot of the number of function evaluations is presented for successful runs
633 from the 30 independent runs during dry, below-normal and wet period years. Although the SC-
634 MGWO algorithm satisfied convergence criteria in the least number of function evaluation, the
635 SC-MGWO was not successful in achieving the optimum solution in many cases. The SP-UCI
636 algorithm is the second fastest method among all the algorithms. The hybrid SC-SAHEL, SC-
637 MFL, and SC-DE are the slowest algorithm for satisfying the convergence criteria, in almost all
638 cases. The slow performance of the hybrid SC-SAHEL is due to the fact that 2 out of 4 (DE and
639 MFL) participating EAs have very slow performance over the response surface. Figure 10
640 demonstrates the number of complexes assigned to each EA during the search, which indicates the
641 dominance of the participating algorithms, and the “award and punishment” logic in the reservoir
642 model. As seen in Figure 10, the MGWO algorithm is dominant in the beginning of the search;
643 although, it is not capable of finding the optimum solution in most cases. The reason for the

644 dominance of the MGWO is the speed of the algorithm in exploring the search space. MGWO is
645 superior to other EAs in the beginning of the search, however, after a few iterations, the MCCE
646 algorithm took the precedence and become the dominant algorithm over other EAs. MGWO and
647 DE are less involved in the rest of the optimization process after the initial steps. However,
648 competition between MCCE and MFL continues. Although contribution of MGWO and DE are at
649 minimum in rest of the optimization process, they are utilizing a part of information within the
650 population. This can affect the speed and performance of the SC-SAHEL algorithm. In both the
651 wet and below-normal cases, the hybrid SC-SAHEL algorithm is mostly terminated by reaching
652 the maximum number of function evolution. However, the mean objective function value obtained
653 by the hybrid SC-SAHEL is still superior to most of the algorithms.

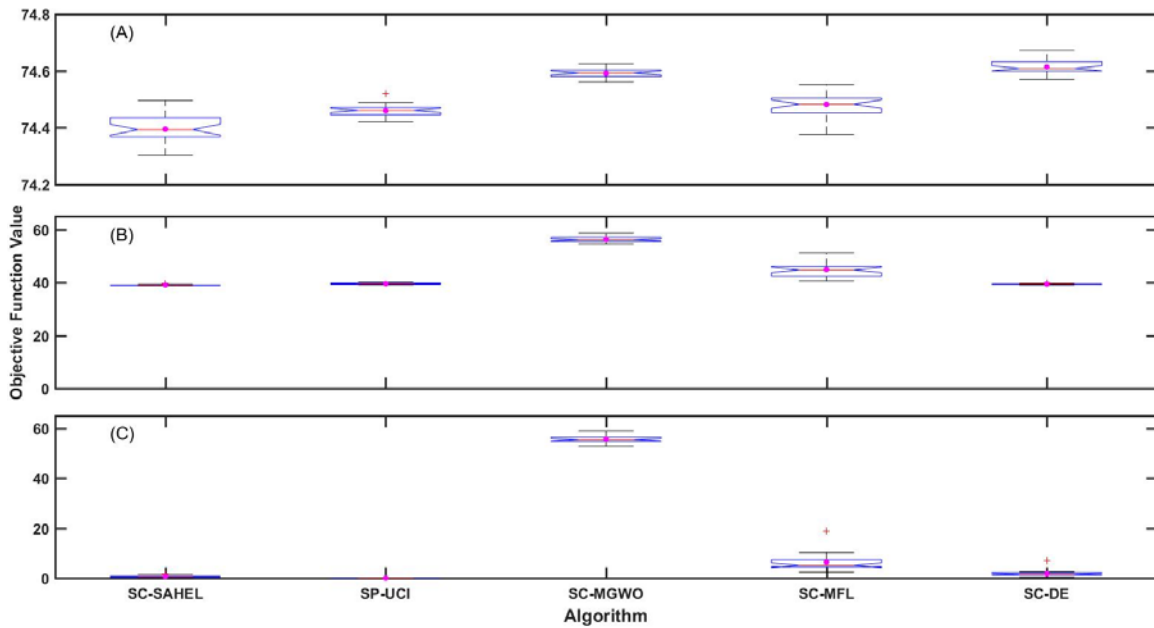
654 The performance of the SC-SAHEL can be affected by the settings of the algorithm.
655 Different settings have been tested and evaluated for the reservoir model. The results show that
656 the number of evolution steps before shuffling can influence the performance of the hybrid SC-
657 SAHEL algorithm.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

667 value smaller than 92. The algorithm is less sensitive to other settings for the reservoir model,
668 however they can still affect the performance of the algorithm.

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

686 The results of the real-world application show the potential of the newly developed SC-
687 SAHEL framework for solving high dimension problems. In general, the hybrid algorithm was
688 more successful in finding a feasible solution in comparison to single-method algorithms. In some
689 cases, the hybrid SC-SAHEL was terminated due to the large number of function evaluations.
690 However, the performance of the hybrid SC-SAHEL is always comparable to the best performing

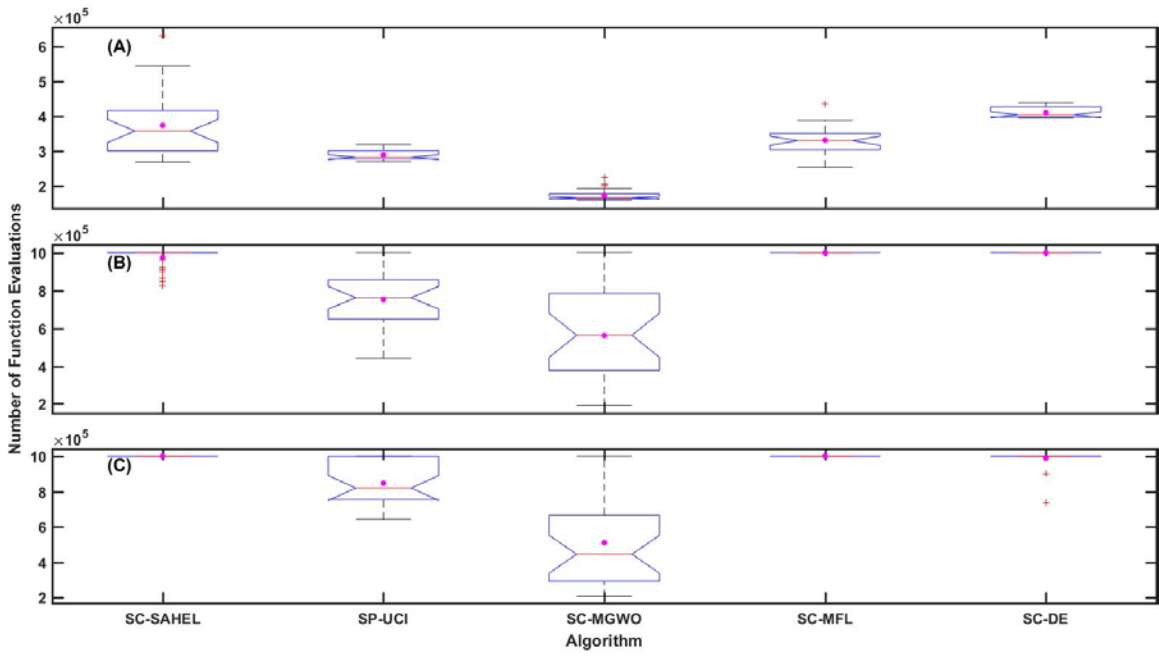
691 method. This shows the potential of the SC-SAHEL for solving a broad class of optimization
692 problems. Besides, the framework provides insight into the performance of the algorithms at
693 different steps of the optimization process. This feature of the SC-SAHEL algorithm can aid the
694 user to select the best setting and EA for the problem.



695

696
697

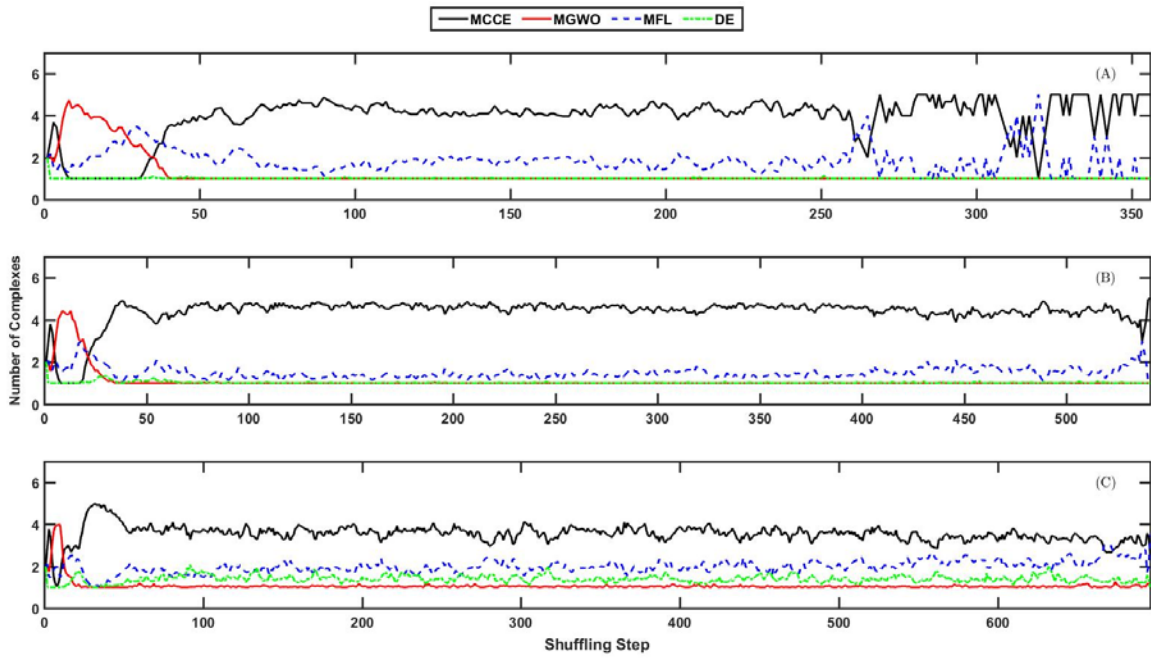
Figure 8. Boxplots of objective function values for successful runs among 30 independent runs, for dry (A), below-normal (B) and wet period (C). The mean of objective functions values is shown with pink marker.



698

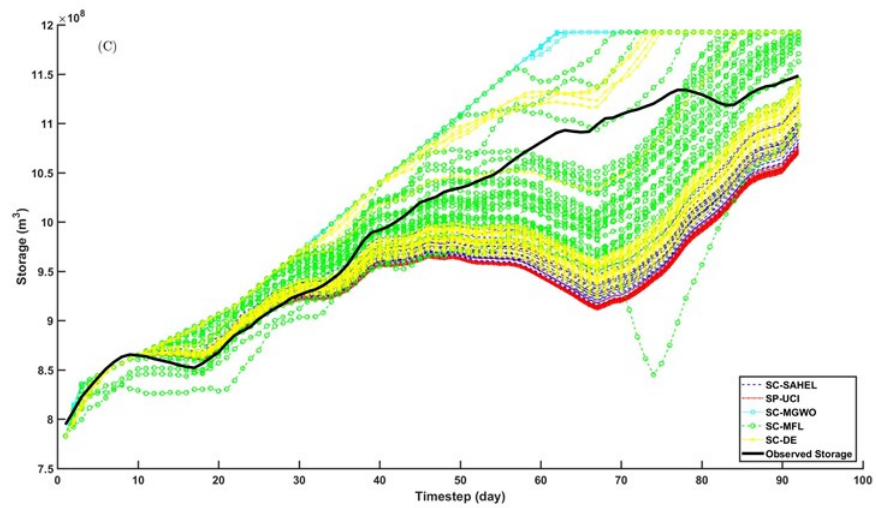
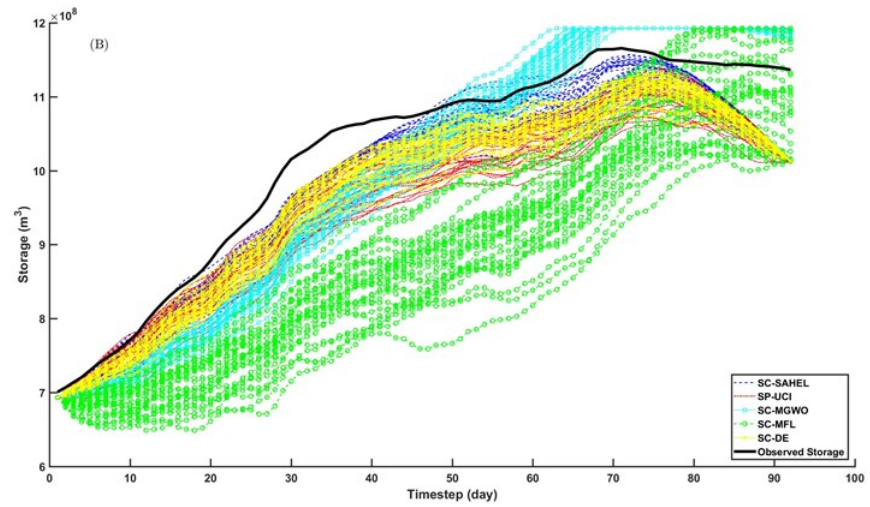
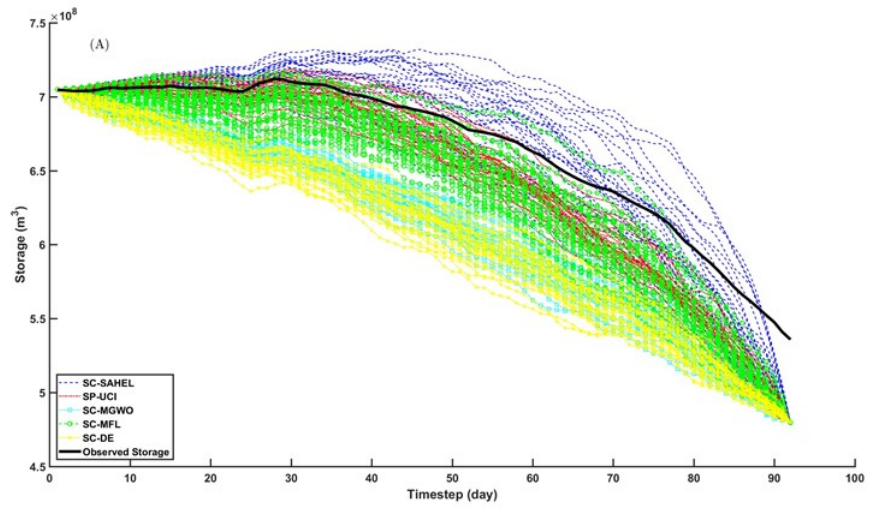
699
700

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



701
702

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



703

704

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

705 **5 Conclusions and remarks**

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

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

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

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

738 **Acknowledgments and data**

739 This work is supported by U.S. Department of Energy (DOE Prime Award # DE-
740 IA0000018), California Energy Commission (CEC Award # 300-15-005), NSF CyberSEES
741 Project (Award CCF-1331915), NOAA/NESDIS/NCDC (Prime award NA09NES4400006 and
742 NCSU CICS and subaward 2009-1380-01), and the U.S. Army Research Office (award W911NF-
743 11-1-0422). The Folsom reservoir bathymetry information used here is provided by Dr. Erfan
744 Goharian from UC Davis, who also helped us for setting up the reservoir model. The authors would
745 like to thank the comments of the four anonymous reviewers which significantly improved the
746 quality of the paper.

747 **References**

- 748 Barati, R., Neyshabouri, S.S. and Ahmadi, G. (2014) Sphere drag revisited using shuffled
749 complex evolution algorithm.
- 750 Beven, K.J. (2011) Rainfall-runoff modelling: the primer, John Wiley & Sons.
- 751 Blum, C. and Roli, A. (2003) Metaheuristics in combinatorial optimization: Overview and
752 conceptual comparison. *ACM Comput. Surv.* 35(3), 268-308.
- 753 Boussaïd, I., Lepagnot, J. and Siarry, P. (2013) A survey on optimization metaheuristics.
754 *Information sciences* 237(Supplement C), 82-117.
- 755 Boyle, D.P., Gupta, H.V., and Sorooshian, S. (2000) Toward improved calibration of
756 hydrologic models: Combining the strengths of manual and automatic methods. *Water Resources*
757 *Research* 36(12), 3663-3674.
- 758 Chu, W., Gao, X. and Sorooshian, S. (2010) Improving the shuffled complex evolution
759 scheme for optimization of complex nonlinear hydrological systems: Application to the
760 calibration of the Sacramento soil-moisture accounting model. *Water Resources Research* 46(9),
761 n/a-n/a.
- 762 Chu, W., Gao, X. and Sorooshian, S. (2011) A new evolutionary search strategy for global
763 optimization of high-dimensional problems. *Information sciences* 181(22), 4909-4927.
- 764 Coello, C.A.C., Lamont, G.B. and Van Veldhuizen, D.A. (2007) *Evolutionary algorithms for*
765 *solving multi-objective problems*, Springer.
- 766 Črepinšek, M., Liu, S.-H. and Mernik, M. (2013) Exploration and exploitation in
767 evolutionary algorithms: A survey. *ACM Computing Surveys (CSUR)* 45(3), 35.
- 768 Ding, Y., Wang, C., Chaos, M., Chen, R. and Lu, S. (2016) Estimation of beech pyrolysis
769 kinetic parameters by Shuffled Complex Evolution. *Bioresource Technology* 200, 658-665.
- 770 Dorigo, M., Maniezzo, V. and Coloni, A. (1996) Ant system: optimization by a colony of
771 cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*
772 26(1), 29-41.
- 773 Duan, Q., Sorooshian, S. and Gupta, V. (1992) Effective and efficient global optimization for
774 conceptual rainfall-runoff models. *Water Resources Research* 28(4), 1015-1031.
- 775 Duan, Q.Y., Gupta, V.K. and Sorooshian, S. (1993) Shuffled complex evolution approach for
776 effective and efficient global minimization. *Journal of Optimization Theory and Applications*
777 76(3), 501-521.

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

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

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

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

786 Eusuff, M.M. and Lansey, K.E. (2003) Optimization of water distribution network design
787 using the shuffled frog leaping algorithm. *Journal of Water Resources Planning and Management*
788 129(3), 210-225.

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

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

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

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

799 Golberg, D.E. (1989) *Genetic algorithms in search, optimization, and machine learning*.
800 Addison wesley 1989, 102.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

832 Liong, S.-Y. and Atiquzzaman, M. (2004) Optimal design of water distribution network
833 using shuffled complex evolution. *Journal of The Institution of Engineers, Singapore* 44(1), 93-
834 107.

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

837 Madsen, H. (2003) Parameter estimation in distributed hydrological catchment modelling
838 using automatic calibration with multiple objectives. *Advances in Water Resources* 26(2), 205-
839 216.

840 Maier, H.R., Kapelan, Z., Kasprzyk, J., Kollat, J., Matott, L.S., Cunha, M.C., Dandy, G.C.,
841 Gibbs, M.S., Keedwell, E., Marchi, A., Ostfeld, A., Savic, D., Solomatine, D.P., Vrugt, J.A.,

842 Zecchin, A.C., Minsker, B.S., Barbour, E.J., Kuczera, G., Pasha, F., Castelletti, A., Giuliani, M.
843 and Reed, P.M. (2014) Evolutionary algorithms and other metaheuristics in water resources:
844 Current status, research challenges and future directions. *Environmental Modelling & Software*
845 62(Supplement C), 271-299.

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

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

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

853 Mirjalili, S., Saremi, S., Mirjalili, S.M. and Coelho, L.d.S. (2016) Multi-objective grey wolf
854 optimizer: A novel algorithm for multi-criterion optimization. *Expert Systems with Applications*
855 47(Supplement C), 106-119.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

900 Vrugt, J.A. and Robinson, B.A. (2007) Improved evolutionary optimization from genetically
901 adaptive multimethod search. *Proceedings of the National Academy of Sciences* 104(3), 708-
902 711.

903 Vrugt, J.A., Robinson, B.A. and Hyman, J.M. (2009) Self-Adaptive Multimethod Search for
904 Global Optimization in Real-Parameter Spaces. *IEEE Transactions on Evolutionary Computation*
905 13(2), 243-259.

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

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

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

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

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

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

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

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

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

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

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

934 Appendix A. Modified Competitive Complex Evolution (MCCE)

935 MCCE algorithm pseudo code is as follow:

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

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

$$939 \quad p = \frac{2(NPT+1-n)}{NPT(NPT+1)}, \quad (A1)$$

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

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

944 *Step 3.* The selected individuals are then stored in \mathbf{S} , forming a simplex. Generate offspring
945 according to following steps.

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

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

$$950 \quad \vec{r} = 2\vec{c} - \vec{w}. \quad (A2)$$

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

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

$$954 \quad \vec{e} = 2\vec{r} - \vec{c}. \quad (A3)$$

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

957 IV. Outside contraction: If $f_d \leq f_r < f_w$, calculate the outside contraction point,
958 $\vec{oc} = \vec{c} + 0.5(\vec{r} - \vec{c})$. (A4)

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

961 V. Inside contraction: If $f_w < f_r$ calculate inside contraction point,
962 $\vec{ic} = \vec{c} + 0.5(\vec{w} - \vec{c})$. (A5)

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

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

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

971 b. Modify \vec{D} as follow

972 $\vec{D}_m = 2(\vec{D} + \text{mean}(\vec{D}))$. (A6)

973 c. Generate a new covariance matrix R' , with \vec{D}_m as diagonal and zeroes everywhere
974 else.

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

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

979 Appendix B. Modified Frog Leaping (MFL)

980 Modified FL (MFL) algorithm is as follow,

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

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

$$984 \quad p = \frac{2(NPT+1-n)}{NPT(NPT+1)}, \quad (B1)$$

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

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

988 *Step 3.* The selected individuals are stored in \mathbf{S} , forming a subcomplex. Generate offspring
989 according to following steps.

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

$$992 \quad \vec{n}_b = \vec{w} + (0.5 \times R + 1.5)(\vec{b} - \vec{w}), \quad (B2)$$

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

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

$$997 \quad \vec{n}_B = \vec{w} + 0.5 \times R(\vec{b} - \vec{w}), \quad (B3)$$

998 Evaluate objective function for the new point and get f_B . If $f_B < f_w$ set the offspring set
999 $\vec{o} = \vec{n}_B$ and go to (IV).

1000 III. Censorship step: If $f_w < f_B$, randomly generate the offspring, \vec{o} by sampling within
1001 the range of individuals in the subcomplex.

1002 **IV.** Replace the worst individual in the complex with the offspring, \vec{o} . Let $i = i + 1$. If $i \leq I$,
1003 go to (Step 1); otherwise sort the complex and return the evolved complex.

1004 Appendix C. Modified Grey Wolf Optimizer (GWO)

1005 Modified Grey Wolf Optimizer is as follow:

1006 *Step 0.* Initialize $i = 1$, and get maximum number of iteration allowed, I .

1007 *Step 1.* Sort the individuals in the order of increasing objective function value. Assign individuals
1008 a triangular probability (except for the fittest point) using following equation:

$$1009 \quad p = \frac{2(NPT+1-n)}{NPT(NPT+1)}, \quad (C1)$$

1010 where NPT is the number of individuals in the complex and n is the rank of the sorted
1011 individuals.

1012 *Step 2.* Select $d+1$ individuals (d is problem dimension) from the complex, with triangular
1013 probability, including the fittest point in the complex and store them in \mathbf{S} .

1014 *Step 3.* Select the best three points in the \mathbf{S} and store them in $\vec{\alpha}$, $\vec{\beta}$ and $\vec{\gamma}$, respectively. The worst
1015 point in the \mathbf{S} , is stored in \vec{w}

1016 *Step 4.* For each of $\vec{\alpha}$, $\vec{\beta}$ and $\vec{\gamma}$, evolve individuals according to the following procedure,

1017 I. Derive \vec{A} and \vec{C} as follow for $\vec{\alpha}$, $\vec{\beta}$ and $\vec{\gamma}$,

$$1018 \quad \vec{A} = 4 \times \vec{r}_1 - 2, \quad (C2)$$

$$1019 \quad \vec{C} = 2 \times \vec{r}_2. \quad (C3)$$

1020 where \vec{r}_1 , \vec{r}_2 are two independent random vectors, which have d dimensions and values in
1021 range of $[0,1)$.

1022 II. Derive \vec{D} , for $\vec{\alpha}$, $\vec{\beta}$ and $\vec{\gamma}$ as follow,

$$1023 \quad \vec{D}_\alpha = |\vec{C}_\alpha \times \vec{X}_\alpha - \vec{w}|, \vec{D}_\beta = |\vec{C}_\beta \times \vec{X}_\beta - \vec{w}|, \vec{D}_\gamma = |\vec{C}_\gamma \times \vec{X}_\gamma - \vec{w}|. \quad (C4)$$

1024 III. Derive \vec{Z} , for $\vec{\alpha}$, $\vec{\beta}$ and $\vec{\gamma}$ as follow,

$$1025 \quad \vec{Z}_\alpha = \vec{X}_\alpha - \vec{A}_\alpha \cdot (\vec{D}_\alpha), \vec{Z}_\beta = \vec{X}_\beta - \vec{A}_\beta \cdot (\vec{D}_\beta), \vec{Z}_\gamma = \vec{X}_\gamma - \vec{A}_\gamma \cdot (\vec{D}_\gamma). \quad (C5)$$

1026 IV. Generate new point by finding the centroid of \vec{Z}_α , \vec{Z}_β and \vec{Z}_γ ,

1027
$$\vec{C} = \frac{\vec{Z}_\alpha + \vec{Z}_\beta + \vec{Z}_\gamma}{3}. \quad (C6)$$

1028 V. Calculate and store objective function value for the new point, f_C . If the new point is

1029 better than the worst point among the selected points, $f_C < f_w$, set $\vec{o} = \vec{C}$, go to step 7.

1030 *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

1031 follow:

1032
$$\vec{A} = 2 \times \vec{r}_1 - 1, \quad (C7)$$

1033 *Step 6.* If the newly generated individual is worse than the worst individuals in subcomplex,

1034 generate a new point with uniform random sampling within the range of individuals in the

1035 complex. Store the new point in \vec{o} .

1036 *Step 7.* Replace the worst individual among selected points in the complex with the offspring, \vec{o} .

1037 Let $i = i + 1$. If $i \leq I$, go to (Step 1); otherwise sort the complex and return the evolved

1038 complex.

1039 Appendix D. Modified Differential Evolution (DE)

1040 Modified differential evolution algorithm is as follow:

1041 *Step 0.* Initialize $i = 1$, and get maximum number of iteration allowed, I .

1042 *Step 1.* Sort the individuals in the order of increasing objective function value. Assign individuals
1043 a triangular probability, using following equation:

$$1044 \quad p = \frac{2(NPT+1-n)}{NPT(NPT+1)}, \quad (D1)$$

1045 where NPT is the number of individuals in the complex and n is the rank of the sorted
1046 individuals.

1047 *Step 2.* Select $d+1$ points (d is problem dimension) from the complex with the assigned
1048 probability and store them along with the fittest point in the complex in \mathbf{S} .

1049 *Step 3.* The selected individuals are sorted and stored in \mathbf{S} , forming a subcomplex. Generate
1050 offspring according to following steps.

1051 I. Generate a new point with the worst point in \mathbf{S} , \vec{w} and using the top three individuals in
1052 the subcomplex,

$$1053 \quad \vec{V}_1 = \vec{w} + 2f(\vec{s}_1 - \vec{w}) + 2f(\vec{s}_2 - \vec{s}_3), \quad (D2)$$

1054 where \vec{w} is the worst point in the \mathbf{S} , \vec{s}_1 , \vec{s}_2 , and \vec{s}_3 are three selected individuals. Then

1055 mutation, and crossover operator is applied to the \vec{w} and \vec{V}_1 to generate \vec{V}_{n1} . The objective
1056 function value for the new point is calculated and stored in f_{n1} . If $f_{n1} < f_w$; set $\vec{o} = \vec{V}_{n1}$
1057 and go to (V).

1058 II. If $f_w < f_{n1}$, generate a new point with the worst point in \mathbf{S} , \vec{w} and using the top three
1059 points in the subcomplex as follow,

$$1060 \quad \vec{V}_2 = \vec{w} + 0.5f(\vec{s}_1 - \vec{w}) + 0.5f(\vec{s}_2 - \vec{s}_3), \quad (D3)$$

1061 After mutation, crossover operator is applied to the \vec{w} and \vec{V}_2 to generate \vec{V}_{n2} . Then, the
1062 objective function for the new point is derived and stored in f_{n2} . If $f_{n2} < f_w$; set $\vec{o} = \vec{V}_{n2}$
1063 and go to (V).

1064 III. If $f_w < f_{n2}$, generate a new point with the worst point in \mathbf{S} , \vec{w} and using the top three
1065 points in the subcomplex as follow,

$$1066 \quad \vec{V}_3 = \vec{w} + f(\vec{s}_1 - \vec{w}) + f(\vec{s}_2 - \vec{s}_3), \quad (\text{D4})$$

1067 After mutation, crossover operator is applied to the \vec{w} and \vec{V}_3 to generate \vec{V}_{n3} . The
1068 objective function value is calculated and stored in f_{n3} . If $f_{n3} < f_w$; set $\vec{o} = \vec{V}_{n3}$ and go to
1069 (V).

1070 IV. If the newly generated point is worse than the worst point in subcomplex, generate a new
1071 point from uniform random distribution within the range of points in the complex. Store
1072 the new point in \vec{o} .

1073 V. Replace the worst point in the complex with the offspring, \vec{o} . Let $i = i + 1$. If $i \leq I$, go to
1074 (Step 1); otherwise sort the complex and return the evolved complex.

- 1075 **Abbreviation**
- 1076 AMALGAM-SO: A Multialgorithm Genetically Adaptive Method for Single Objective
- 1077 Optimization
- 1078 CCE: Competitive Complex Evolution
- 1079 DE: Differential Evolution
- 1080 EA: Evolutionary Algorithm
- 1081 EMP: Evolutionary Methods Performance
- 1082 FL: Frog Leaping
- 1083 GWO: Grey Wolf Optimizer
- 1084 LHS: Latin Hypercube Sampling
- 1085 MCCE: Modified Competitive Complex Evolution
- 1086 MFL: Modified Frog Leaping
- 1087 MGWO: Modified Grey Wolf Optimizer
- 1088 MOCOM-UA: Multi-Objective Complex evolution, University of Arizona
- 1089 MOSCEM: Multi-Objective Shuffled Complex Evolution Metropolis
- 1090 NFL: No Free Lunch
- 1091 PCA: Principal Component Analysis
- 1092 PSO: Particle Swarm Optimization
- 1093 SaDE: Self-adaptive DE algorithm
- 1094 SCE-UA: Shuffle Complex Evolution-developed at University of Arizona
- 1095 SCEM-UA: Shuffled Complex Evolution Metropolis algorithm-developed at University of
- 1096 Arizona
- 1097 SC-SAHSL: Shuffle Complex-Self Adaptive Hybrid EvoLution

- 1098 SP-UCI: Shuffled Complex strategy with Principal component analysis-developed at University
- 1099 of California, Irvine
- 1100 URS: Uniform Random Sampling