Knowledge-Based Systems 109 (2016) 174–186

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

A dynamic optimization approach to the design of cooperative co-evolutionary algorithms

Xingguang Peng[a,*], Kun Liu[a], Yaochu Jin[b]

[a] School of Marine Science and Technology, Northwestern Polytechnical University, Xi'an, Shaanxi, 710072, PR China
[b] Department of Computer Science, University of Surrey, Guildford, GU2 7XH, United Kingdom

ARTICLE INFO

Article history:
Received 5 April 2016
Revised 30 June 2016
Accepted 1 July 2016
Available online 2 July 2016

MSC:
00-00-00

Keywords:
Cooperative co-evolutionary algorithm
Dynamic landscapes
Multi-population mechanism

ABSTRACT

Cooperative co-evolutionary algorithm (CCEA) decomposes a problem into several subcomponents and optimizes them separately. This divide-and-conquer feature endows CCEAs with the capability of distributed and high-efficiency problem solving. However, traditional CCEAs trend to converge to Nash equilibrium rather than the global optimum due to information loss accompanied with problem decomposition. Moreover, the interactive nature makes the subcomponents' landscapes dynamic, which increases the challenge to conduct global optimization. To address these problems, a multi-population mechanism based CCEA (mCCEA) was proposed to compensate information in dynamic landscapes. The mCCEA is decentralized for each subcomponent since it doesn't need centralized archive or information sharing. It focuses on both the global and the local optima of each subcomponent by maintaining multiple populations and conducting local search in dynamic landscapes. These optima are seen as the current representatives of the subcomponents and used by the other subcomponents to construct their complete solutions for fitness evaluation. Experimental study was conducted based on a wide range of benchmark functions. The performance of the proposed algorithm was compared with several peer algorithms from the literature. The experimental results show effectiveness and advantage of the proposed algorithm.

© 2016 Elsevier B.V. All rights reserved.

1. Introductions

Cooperative co-evolution is primarily a biological concept, but has been applied to many other fields by analogy. In the field of evolutionary computation, Potter and De Jong [1,2] proposed cooperative co-evolutionary algorithm (CCEA) by introducing divide-and-conquer strategy to divide a problem into several subcomponents which are then evolved separately. The divide-and-conquer evolution scheme offers CCEAs the advantage of decomposing a complex optimization problem into a number of relatively simpler subproblems and solving them concurrently. This endows CCEAs with the potential of distributed computation and high problem-solving efficiency. CCEAs have been extended to fields of machine learning [3], rough set attribute reduction [4], multi-objective optimization [5],wireless sensor networks [6], cooperative planning [7] and clustering [8]. Especially in recent years, CCEAs have been applied to deal with large scale optimization based on many successful problem decomposition approaches [9–12].

However, the divide-and-conquer strategy is a double-edged sword. By decomposing a problem into subcomponents, CCEAs may lose a great deal of information. A subpopulation only represents the corresponding segments of the complete solutions. Collaborators must be collected from the other subpopulations to construct complete solutions for fitness evaluation. Obviously, the fitness of an individual is sensitively affected by the collaborators from the other components. In addition, CCEAs can be modeled with evolutionary game theory (ETG) [13]. From the theory perspective, CCEAs may gravitate towards suboptimal solutions represented by Nash equilibria rather than global optima in the complete problem. This may lead to some inherent problems. The most typical one is relative overgeneralization (RO), in which the subpopulations are likely to converge to the Nash equilibria with larger basins of attraction regardless of whether they are global optima (as illustrated in the next section).

The suboptimal convergence of simple CCEAs is essentially different from that of the common EAs, which can be addressed by maintaining proper population diversity. Effective information compensation methods should be incorporated into CCEAs to enhance global optimization. Many works have been done to design information compensation scheme for CCEAs. From the interactive

* Corresponding author.
E-mail addresses: xgpeng.nwpu@gmail.com (X. Peng), liukunkmz@126.com (K. Liu), yaochu.jin@surrey.ac.uk (Y. Jin).

http://dx.doi.org/10.1016/j.knosys.2016.07.001
0950-7051/© 2016 Elsevier B.V. All rights reserved.

nature of the CCEAs, information compensation could be archived from both sending and receiving perspectives.

The archive based method is an intuitive and popular way to compensate information from the sending perspective. For example, in [14,15] Nash memory is maintained to help subcomponents converge to the global optimal equilibrium. Also, the nature of diversity maintenance of Pareto dominance population [16] or reference [17] is utilized to design CCEAs for global optimization. In addition to explicitly recording information in additional archives, the co-evolutionary information can also be recorded or maintained implicitly. In [18], memory is implicitly added to a co-evolutionary computation framework by embedding the subpopulations into a spatial geometry, which can help maintain adaptive gradients to improve the global optimization performance.

As for the research from sending perspective, it has not been studied as extensively as the way to compensate from receiving perspective. Most of the work in the literature uses a *best-and-N-random* strategy (one best and N random individuals), and little work can be seen to explore what information should be exchanged among subcomponents to achieve information compensation. Panait and Luke [19,20] argued that "subpopulations should not necessarily explore only their most promising solutions, but also those solutions that provide the other subpopulations with accurate projections of the joint search space." They proposed a scheme to help one subcomponent select informative collaborators for the co-evolution of the other subcomponents. Although the experimental results are positive on the RO-featured problems, one subpopulation needs to access all individuals of the others'. Such centralized whole-population accessing will lead to large amount of communication and fitness evaluations, which is impractical especially in network based cases.

Therefore, the research into CCEAs which can globally optimize with reasonable numbers of collaborators is still in its infancy, and the need for efficient methodologies is demanded. In this paper, we design an improved CCEA under the following intuitive motivation: Local and global optima can well feature the landscapes of subcomponents, when these optima are used as representative collaborators subpopulations will be provided with more sufficient information for co-evolution.

To search those representative collaborators, the dynamic nature of the landscapes of subcomponents must be taken into account. The evaluation of an individual depends on the collaborators from the other subpopulations and these collaborators may vary through out the co-evolution process, which means that the evaluation of a certain individual is not static but dynamic. Although it has been realized in early work [13], little work deals with it. If this feature were considered when designing a CCEA, more representative information could be searched and the subpopulations could conduct co-evolution with better collaborators. Fortunately, the methodologies for enhancing EAs to search in dynamic landscapes have been intensively studied especially in the past decade (see the surveys and books [21–24]). Thanks to these improvements, co-evolution of subcomponents regarding the dynamic landscapes has become practical.

Bearing these ideas and motivations in mind, an improved CCEA is suggested, investigated, and discussed in this work. In general, the contributions of this work can be summarized as follows:

- A multi-population scheme has been proposed to dynamically discover and maintain multiple optima for a given subcomponent: several child populations can be adaptively split off from the base population (the population of a given subcomponent) or merge to track the spatial optima (local or global). The base population adopts a population based genetic algorithm to explore new optimum in the whole sub-landscape, while the child populations adopt local search to exploit discovered optima efficiently.
- A two-fold information compensation technique is introduced: The optima found by the multi-population scheme are used as information carriers and exchanged among subcomponents to achieve information compensation from the interaction sending perspective. On the other hand, historic best solutions of the child populations are maintained along with the complete solution contexts. Such solution contexts are utilized in the fitness evaluation procedure to achieve information compensation from the interaction receiving perspective. Such a two-fold information compensation mechanism enables CCEA to effectively tackle the inherent difficulty when optimizing RO-featured problems.
- The proposed multi-population scheme based CCEA framework can run without centralized information sharing or large amount of random information exchange, which make the algorithm more practical for real-world applications.

This paper is extended from our previous work [25] in which a dynamic multi-population evolutionary algorithm and a grid-based archive were integrated into CCEA to conduct information compensation. This paper extends this previous work in the following aspects. First, we modified the criterion for merging two overlapped populations; Second, we use different optimizers in different kind of populations; Third, we proposed a new method to conduct fitness evaluation using the historic information and the collaborators instead of grid-base archive method. At last, in experiments section, wider benchmark functions and more peer algorithms are involved, comparison and analysis are more exhaustive.

The remainder of this paper is organized as follows. In Section 2, the CCEA principle and the relative overgeneralization problem is demonstrated. The techniques concerning evolutionary computation in dynamic landscapes are also briefly reviewed. Section 3 is devoted to the description of the proposed algorithm. Section 4 presents the experimental results and discussions. Finally, Section 5 provides conclusions and future work.

2. Background and related work

2.1. CCEA and relative overgeneralization problem

CCEAs speed up the optimization process by decomposing a problem into subcomponents that can be concurrently optimized by several subpopulations. The main difference between classical evolutionary algorithms (EAs) and CCEAs is that in CCEAs an individual only codes a segment (according to its subcomponent) of the solution, it have to collect collaborators from the other subpopulations to construct complete solutions for fitness evaluation. In the example shown in Fig. 1, the whole problem is divided into three subcomponents and each of them is evolved by a subpopulation. As for the individual "0101" in subpopulation A, its fitness evaluation is realized by combining with collaborators from the other subpopulations ("1001" from subpopulation B and "1100" from subpopulation C). Consequently, the fitness value of "0101" is indeed the evaluation value of "0101|1001|1100" according to the global fitness function f.

It is always highly desired that a given problem should be decomposed according to the dependence relationship between the decision variables when conducting cooperative co-evolution. If any decision variable in a subcomponent only depends on the variables within the subcomponent, the algorithm can effectively converge to the global optimum by exchanging the best solutions among subcomponents. In other words, if there is no information loss after decomposition, global optimization could be guaranteed. This is because the underlying landscape of a subcomponent is not

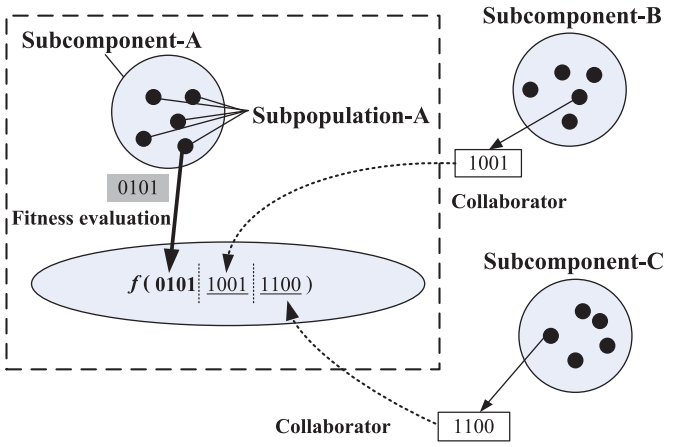


Fig. 1. Illustration of the fitness evaluation in the CCEAs (developed from [13]).

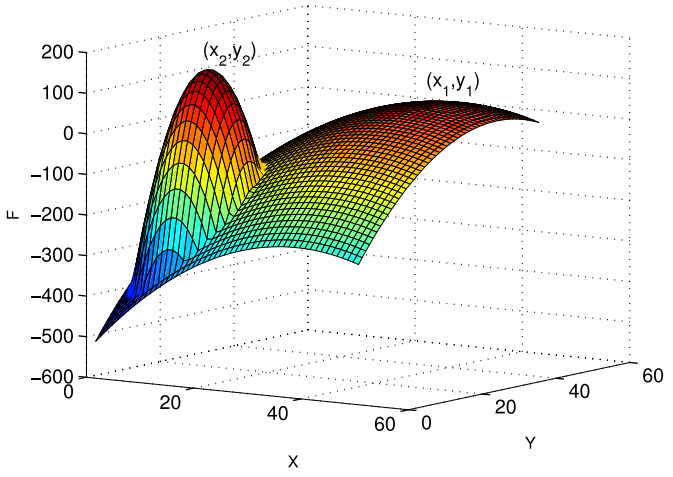


Fig. 2. Landscape of a typical RO-featured landscape.

affected by the evolution of the decision variables in the other subcomponents.

However, to obtain an ideal decomposition is very difficult for some complex problems or even impractical for some real-world applications. Given decomposition errors, the fitness calculation is prune to errors too. The collaborators provided by the other subpopulations may significantly affect the evaluation. If the collaborators can properly reflect the features of their subcomponents, the subpopulations may cooperatively converge to the global optimum. Unfortunately, traditional CCEA lacks effective ways to search good representative information of subcomponents. Without being provided with good representative collaborators, an subpopulation cannot be properly evaluated and the co-evolution may be misguided. This will cause some inherent problems among which *relative overgeneralization* (RO) problem may be the most typical one. The word *relative overgeneralization* means the use of small and/or non-representative samples of real data to make an inference that is incorrect.

As seen in Fig. 2, the RO problem can be demonstrated where subpopulations are likely to be attracted to the larger basin area where there are many strategies that perform well and finally converges to local optimum (x_1, y_1) rather than the global optimum (x_2, y_2). This local convergence is different to premature convergence of classic EAs. With proper diversity maintaining methods, classic EAs can escape from local optima and theoretically converge to the global optimum. In contrast, it has been theoretically and experimentally analyzed that even with infinite population and no probabilistic noise, traditional CCEAs are still not necessarily to converge to the global optimum [13].

2.2. EAs in dynamic landscapes

The main challenge of traditional EAs for solving DOPs is how to properly maintain the population diversity to keep algorithms with efficient exploring ability in dynamic environment whose change is detectable or non-detectable. To address this issue, over the past two decades, a number of researchers have developed many methods to maintain diversity for traditional EAs to continuously adapt to the changing environment. Most of these methods can be categorized into the following five types of approaches:

(1) Increasing the diversity after a change: This kind of algorithms aim to handle convergence directly [26–28]. These methods are more suitable for problems with highly frequent changes and the changes are not very big.

(2) Maintaining the diversity throughout the run: This kind of algorithms do not maintain diversity with an explicit change detection, instead they compensate diversity through the run [29–32]. In contrast to the first category, these algorithms are more suitable for problems with severe and rare changes.

(3) Memory based approaches: This kind of algorithms record historic information with implicit memory [33–35] or explicit archives [36–40]. According to the experimental results of the literatures, memory based approaches commonly perform well in cycling landscapes.

(4) Prediction based approaches: This kind of algorithms predict the optima in new environment from the historic information so as to guide evolutionary search and re-initiate individuals accordingly [41–43]. These algorithms run effectively under the hypothesis that the changes of landscapes are predictable.

(5) Multi-population approaches: This kind of algorithms concurrently maintain several subpopulations to track and discover local optima adaptively. Pioneer methods include shift balance GA [44], self-organizing scouts GA [21], multi-national GA [45]. The ideas of these algorithms have been adopted to develop many population based search algorithms for dynamic landscapes. For example, multi-swarms methods [46], speciation PSO [47], cluster based PSO [48], adaptive multi-population artificial bee colony algorithm [49], multi-nation differential evolution [50]. To some extent, multi-population approaches can be seen a combination of the first three categories and there is no special hypothesis for the dynamics of landscapes.

Overall, multi-population approaches seem to be the most flexible. Moreover, these approaches can search or track the moves of multiple optima, which is helpful for CCEAs' subpopulations to search representative collaborators.

3. Multi-population mechanism based CCEA

3.1. The framework

In this paper, we propose a multi-population mechanism based CCEA (mCCEA). The illustration of the framework of the mCCEA is given in Fig. 3, suppose a problem is decomposed into N subcomponents and each of them is optimized by a corresponding subpopulation. By sending and receiving collaborators with each other, all subpopulations cooperatively coevolve to find the global optimum.

Considering the interactive nature of CCEAs, each subpopulation plays as both sender and receiver of collaborators. As seen in Fig. 3, when sending collaborators, a subpopulation provides its representative collaborators to the other subpopulations. To find such representative collaborators, the subpopulation needs to simultaneously search multiple optima (local or global) rather than single

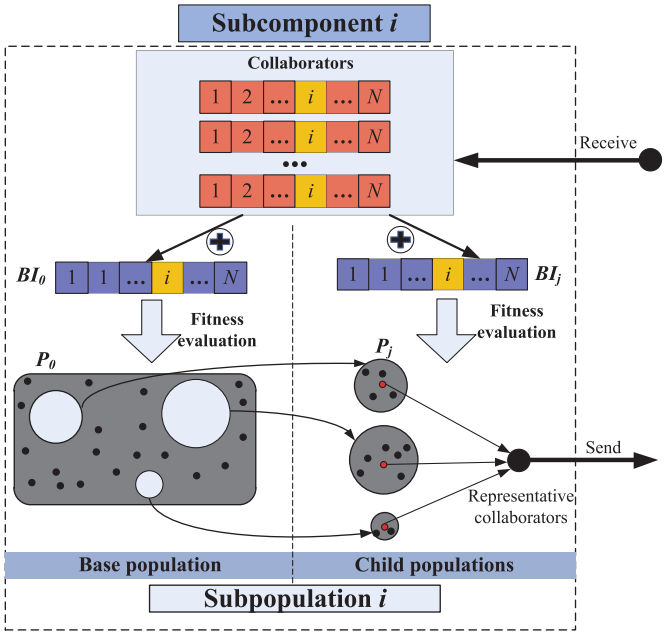


Fig. 3. Framework of proposed algorithm.

optimum. To achieve this, inspired by the idea of multi-population approaches for dynamic landscapes [21], a multi-population mechanism is proposed in this paper. Within this mechanism, the ith subpopulation is composed of one base population P_0 and N_c child populations P_j, $j = 1, \ldots, N_c$. These child populations are adaptively split off from the base population to track the current global or local optima and the remaining population (base population) is used to explore new arising optima. While the base population searches using conventional GA, the child populations conduct local search within their own search spaces. When a child population converges, most of its individuals will be removed to the base population, which provides the base population with more resource to generate extra child populations. This is beneficial when dealing with multimodal optimization with a huge number of optima.

On the other hand, from a receiver's perspective, the collaborators collected from the other subpopulations are combined with individuals of ith subpopulation to construct complete solutions for fitness evaluation. In addition, the historic best individuals of the base population and child populations (denoted by BI_0 and BI_j respectively) are recorded along with the complete solution context. This complete solution context is also used to construct complete solution when conducting fitness evaluation. By maintaining these solution contexts, historic co-evolutionary information could be used when using the collaborators. Note that the fitness of an individual is estimated using the *best-of-N* strategy, i.e. the best fitness value calculated by combining the individual with collaborators or complete solution context.

In summary, in the proposed framework, each subpopulation is evolved separately by a multi-population mechanism. Our framework can run without centralized information sharing or large amount of random information exchange, which make it more practical for real-world applications. In addition, due to sending representative collaborators (current global or local optima) and maintaining historic solution context, information compensation could be expected to help CCEAs avoid inherent problems caused by information loss.

3.2. The multi-population mechanism

In this paper, we propose a multi-population mechanism for each subpopulation to search representative collaborators (i.e. multiple optima) in dynamic landscapes. Inspired by Self-Organizing Scout (SOS) algorithm [21], our method utilizes multiple populations to continually track multiple optima and find newly arisen peak from the dynamic landscapes. Considering the fact that a split-off search space is relatively small, we further integrated local search into our mechanism. In general, there are five steps in the multi-population mechanism which are shown in Algorithm 1.

Algorithm 1 The pseudo code of the multi-population mechanism used by each subpopulation.

1: **while** *termination condition* of co-evolutionary process is not satisfied **do**
2: Check forking criteria and create new child populations;(see Section 3.2.1)
3: Reproduction of offsprings of the base population and child populations; (see Section 3.2.2)
4: Management of child populations; (see Section 3.2.3)
5: **if** at collaborator exchanging iteration **then**
6: Send representative collaborators; (see Section 3.2.4)
7: **end if**
8: Recycle individuals discarded in step 4; (see Section 3.2.5)
9: **end while**

In the following context we will describe how these steps work respectively.

3.2.1. Checking for forking and creating new child populations

In this paper, we use the concept of forking populations which was proposed in [51] and extended in [21] to adapt to dynamic landscapes. The search space of a subcomponent is searched by a base population $P(0)$ and Nc child populations $P(j)$, $j = 1, \ldots, N_c$. When $P(0)$ converges to certain degree a forking process will be triggered: a number of mostly converged individuals are split off from the $P(0)$ and a new child population $P(Nc+1)$ is created accordingly. After forking process, the rest individuals in base population $P(0)$ are randomly re-initiated. Afterwards, the child populations $P(j)$ keep evolving to track multiple optima and the base population $P(0)$ searches from scratch to find new optima. The pseudo code of checking for forking and creating new child populations is given in Algorithm 2 where P_{init} denotes initial population size of a child populations, $r(j)$ and r_{max} denote the radius of jth population's search space and its maximal value respectively, P_{0min} denotes the minimal size of base population, $BI(0)$ and $bestI(0)$ denote the historic best individual and current best individual of $P(0)$ respectively.

Note that the forking criteria of our multi-population mechanism is different from that of the original SOS [21]. When measuring the converging degree of a base population, we only use the stopping criterion used in ordinary genetic algorithms thanks to maintaining historic best individuals their solution context (see in Section 3.3). With such solution context, the fitness value of the best individual varies monotonously rather than arbitrarily. Accordingly, the criterion such as the fitness of best fitness has not increased for a number of generations could be practical in dynamic landscapes. In addition, a density criterion $\sum_{i=1}^{|P(0)|} D(i)$ is used rather than a number of restrictions in SOS.

3.2.2. Reproduction of offsprings for base and child populations

In the mCCEA, base population P_0 and child populations P_j, $j = 1, \ldots, N_c$, run independently within their own search spaces. The offsprings of the base population are not expected to fall into the search spaces of child populations. If any individual in the base population falls into the search space of child populations, it will be given an enormous penalty. In addition, the base population

Algorithm 2 The pseudo code of checking for forking and creating new child populations.

```
isForking = false;
if BI(0) has stopped being improved for at least N_f iterations then
  for each individual I(i) in P(0) do
    if ||I(i) − bestI(0)|| ≤ r_max then
      D(i) = 1;
    else
      D(i) = 0;
    end if
  end for
  if Σ_{i=1}^{|P(0)|} D(i) ≥ P_init and |P(0)| − P_init ≥ P_0min then
    isForking = true;
  end if
end if
if isForking = true then
  Nc = Nc + 1;
  Move the best P_init individuals in P(0) to P(Nc);
  r(Nc) = r_max;
  for each individual I remaining in P(0) do
    repeat
      randomly set I;
    until I dose not fall into any P(j), j = 1, ..., Nc
  end for
end if
```

and child populations evolve in different ways. As for base population, it uses traditional GA with simulated binary crossover and mutation. As for child populations, they conduct the Simplex local search [52] in their relatively small search spaces. If a child population has converged, i.e. the best fitness has not increased for N_f generations and the radius of its search space is smaller than or equal to r_{min}, only two individuals (the best one and a randomly selected one) are kept in the child population and the other individuals are removed into recyclable archive for retrieving back to the base population in the next cycle.

3.2.3. Management of child populations

In order to make the child populations' search spaces shrinks gradually, we introduce a control parameter called shrinking factor f_r from the original SOS. Considering the jth child population $P(j)$, the radius (denoted by $r(j)$) of its search space decreases iteration by iteration with respect to the generation number $g(j)$ and f_r as follows [21]:

$$r(j,t+1) = r_{min} + (r(j,t) - r_{min})^{\frac{g(j)}{-f_r}} \tag{1}$$

The decrease of $r(j)$ will lead to the fact that some individuals may locate outside the new search space. These individuals will be discarded and stored in a recyclable archive Ψ subjected to the minimal population size P_{smin}.

The center $c(j)$ of the search space of child population $P(j)$ is the best individual of $P(j)$. During the search procedure of $P(j)$, the center $c(j)$ may change its location, which will lead the movement of $P(j)$'s search space. This may cause overlapping of adjacent search spaces. The overlapping is usually allowed except that two adjacent child populations are two close to each other. As seen in Fig. 4, overlapping is allowed in case (a) and case (b) while in case (c) overlapping is not allowed and both child populations are needed to be merged together.

When merging two child populations, denoted by $P(1)$ and $P(2)$ respectively, we firstly make a comparison between them. In this paper, the quality of a child population is characterized by its historic best individual. Then, the population with worse historic best individual (suppose $P(2)$) will be merged into the better one, i.e. $P(1)$. More particularly, individuals in $P(2)$ that fall in the search space of $P(1)$ will be moved into $P(1)$ and the reset individuals are discarded and moved to a recyclable archive Ψ. Note that, if the size of $P(1)$ exceeds the maximal child population size P_{smax}, only the best $p_{smax} - |P(1)|$ individuals will be merged to $P(1)$ and the rest individuals are also moved to Ψ. At last, like the SOS, the new radius of $P(1)$ is updated as follow.

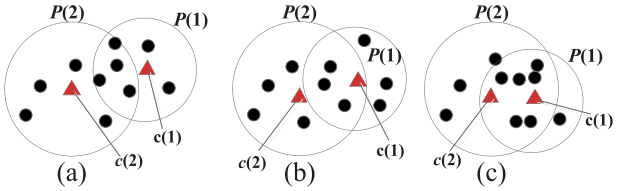


Fig. 4. Illustration of overlapping of two adjacent child populations.

$$r_1 = min\left(\sqrt{r(1)^2 + r(2)^2}, r_{max}\right) \tag{2}$$

3.2.4. Sending representative collaborators

Considering the challenge of dynamic landscapes, in addition to the multiple optima that have been found, the mCCEA also allows exchanging randomly selected individuals to improve the diversity of collaborators. To control the amount of communication, we introduce a control parameter N_{com}. When exchanging collaborators each subpopulation will send N_{com} individuals to the other subpopulations. The best individual of the child populations have the priority. That is, if $N_c \le N_{com}$ each child population's best individual plus $N_c - N_{com}$ randomly selected individuals will be sent. Otherwise, only the best N_{com} individuals will be sent.

3.2.5. Reusing the individuals in recyclable archive

It can be seen that some individuals may be discarded and stored in the recyclable archive Ψ during the following procedures:

- Moving and shrinking of child populations: the individuals that fall outside of the searching spaces of child populations.
- Merging of two overlapped child populations: some individuals of the worse population.

To reuse the resource, the archived individuals are randomly re-initialized and retrieved into the base population $P(0)$ at the end of an every cycle. Note that, the re-initialization of an individual might be done more than once to guarantee that this individual will not locate in the search space of any child population.

3.3. Constructing complete solutions for fitness evaluation

Suppose an m-dimensional optimization problem is decomposed into N subcomponents and each of them is evolved by a subpopulation. The ith subpopulation is composed of one base population and Nc_i split-off child populations which are denoted by $P(j)$, $j = 0, \ldots, Nc_i$ ($j = 0$ indicates the base population). The historic best individuals of $P(j)$ are denoted by $BI_i(j)$, $dim(BI_i(j)) = m$. $Ncom$ collaborators are collected from each subcomponent, i.e. $|Colb_k| = Ncom$.

The Pseudo-code of constructing complete solutions for fitness evaluation is shown in Algorithm 3. In each generation, as for ith subcomponent, the received collaborators and the historic best individuals of base population and child populations are combined to build complete solutions for fitness evaluation. When conducting fitness evaluation an individual should not only be combined with the collaborators from the other subpopulations but also be

Algorithm 3 The pseudo code of constructing complete solutions for fitness evaluation of ith subpopulation.

```
Shuffle collaborators in each {Colb_k}, k ∈ N, k ≠ i;
for l = 1, ..., Ncom do
    Construct collaborative solutions S_i(l) = {Colb_k(l)}, k ∈ N, k ≠ i;
end for
for j = 0, ..., Nc_i do
    Add historic best individual BI_i(j) to S_i(l), i.e. S_i(Ncom + 1) = Genotype(BI_i(j), g), g ∈ m but gth decision value ∉ ith subcomponent;
    for Each individual I in P(j) do
        Complete solution CS(l) = {S(l) ∪ I};
        Fitness(I)=best{F(CS_i(l))}, l ∈ Ncom + 1;
        l_best = l s.t. F(CS_i(l)) is the best;
        if Fitness(I) is better than Fitness(BI_i(j)) then
            BI_i(j) = CS(l_best);
        end if
    end for
end for
```

combined with the complete solution context of the historic best individuals. After calculating fitness values of these complete solutions, the final fitness of this individual is estimated using *best-of-N* strategy, i.e. the maximal (for maximization problem)or minimal (for minimization problem) value. Meanwhile the historic best individuals are also updated according to the fitness evaluation.

3.4. Implementation

The flowchart of each subpopulation is shown in Fig. 5. In the main cycle, at every generation the forking criteria are checked to split off child populations. If forking criteria are met, one child population is split off from the base population. Then the next generation offsprings of base population and existing child populations are generated by genetic operators and the Simplex local search representatively. Base on the new offsprings, the management of child population are processed. This includes search space adjustment (center and radius) and merging child populations that are too close to each other. In the management process some individuals may be removed to recyclable archive and will be reused in the next cycle. At last, fitness evaluation of base and child populations is conducted using the solution context from collaborators and historic best individuals. The best individuals of base population (when no child population exists) or child populations are selected as representative collaborators and sent to the other subpopulation when needed.

4. Experimental study

In this section, two groups of experiments are carried out. The object of the first group of experiments is to compare mCCEA with several peer CCEAs taken from the literature on RO-featured problems and ten benchmark functions on continuous optimization used in a CEC2015 competition. In the second group of experiments, we focus on the mCCEA's sensitivity to different configurations.

4.1. Benchmark problems

4.1.1. RO-featured problems

Since the main objective of this paper is to improve the ability to avoid inherent problems (e.g. the RO problem), we use a class of problem domains called the maximum of two quadratics (MTQ) which have been used to test the global optimization ability of CCEAs in literature [13,16,18–20,53]. These problems include a global optimum and a local suboptimum, where the suboptimum covers a much wider range of the search space and is thus difficult to escape for traditional CCEAs.

The joint reward function for the MTQ class is defined as [18]:

$$MTQ(x_1, x_2) \leftarrow \max \begin{cases} H_1 * \left(1 - \dfrac{16 * (x_1 - X_1)^2}{S_1} - \dfrac{16 * (x_2 - Y_1)^2}{S_1}\right) \\ H_2 * \left(1 - \dfrac{16 * (x_1 - X_2)^2}{S_2} - \dfrac{16 * (x_2 - Y_2)^2}{S_2}\right) \end{cases} \quad (3)$$

where x_1 and x_2 may take values ranging between 0 and 1. H_1, H_2, X_1, Y_1, X_2, Y_2, S_1, and S_2 are parameters that affect the difficulty of the problem domain. More particularly, H_1 and H_2 affect the heights of the two peaks; S_1 and S_2 affect the area that the two peaks cover: a higher value may result in a wider coverage of the specific peak, which makes the algorithm more likely converge to this peak, even though it may be suboptimal; X_1, Y_1, X_2, and Y_2 characterize the locations of the centers of the two quadratics, which affects the relatedness of the two peaks.

In the following experiments, we use the following default parameter settings: $H_1 = 50$, $X_1 = 0.75$, $Y_1 = 0.75$, $X_2 = 0.25$, $Y_2 = 0.25$, $S_1 = 1.6$. H_2 and S_2 is set to 70, 150, 300 and 1/32, 1/64, 1/128 respectively so as to gain a series of test problems with different difficulty. $[X_2, Y_2]$ is the global optimum with highest peak but small covering area while $[X_1, Y_1]$ is the local optimum with a large covering area as shown in Fig 2. For traditional CCEAs, the subpopulations are likely to converge to $[X_1, Y_1]$ where the they are more likely to perform well and reach a Nash equilibrium although such Nash equilibrium is not necessary a global optimum.

4.1.2. Continuous optimization problems

To verify the performance of mCCEA on a wide range of problems, we introduce several continuous optimization benchmark problems used in CEC2015 competition on 'learning-based real-parameter single objective optimization' [54]. Ten functions, i.e. F_1, F_2, F_3, F_4, F_5, F_9, F_{11}, F_{12}, F_{14} and F_{15} are considered in this paper. All of these functions are non-separable with the search range in $[-100, 100]$. For convenience in this paper, we rename these functions as F_1, F_2, ..., F_{10} respectively.

4.2. Algorithms for comparison

To verify the effectiveness of mCCEA, the following CCEAs are used as peer algorithms:

4.2.1. Traditional CCEA (tCCEA)

tCCEA [55] evaluates the fitness of an individual as the best value when combined with collaborators from the other subpopulations: K chosen at random plus the current fittest individual of the other subpopulations. In the following experiments K is set to 3, which means 4 collaborators are sent from one subpopulation to the others when exchanging collaborators.

4.2.2. Biased CCEA (bCCEA)

bCCEA [18] is developed from tCCEA. The fitness of an individual is partly biased according to an algorithmic parameter δ on the reward obtained when collaborating with the collaborators from the other subpopulations like in tCCEA (the best fitness when collaborating with K randomly chosen and one fittest individuals of the other subpopulations). The remaining part of the fitness (i.e. $1 - \delta$) is based on the estimated optimal collaborator (the best historic collaborator). For fair comparison, here K is also set to 3. The biasing ration δ starts with 1, decreasing linearly until reaching at

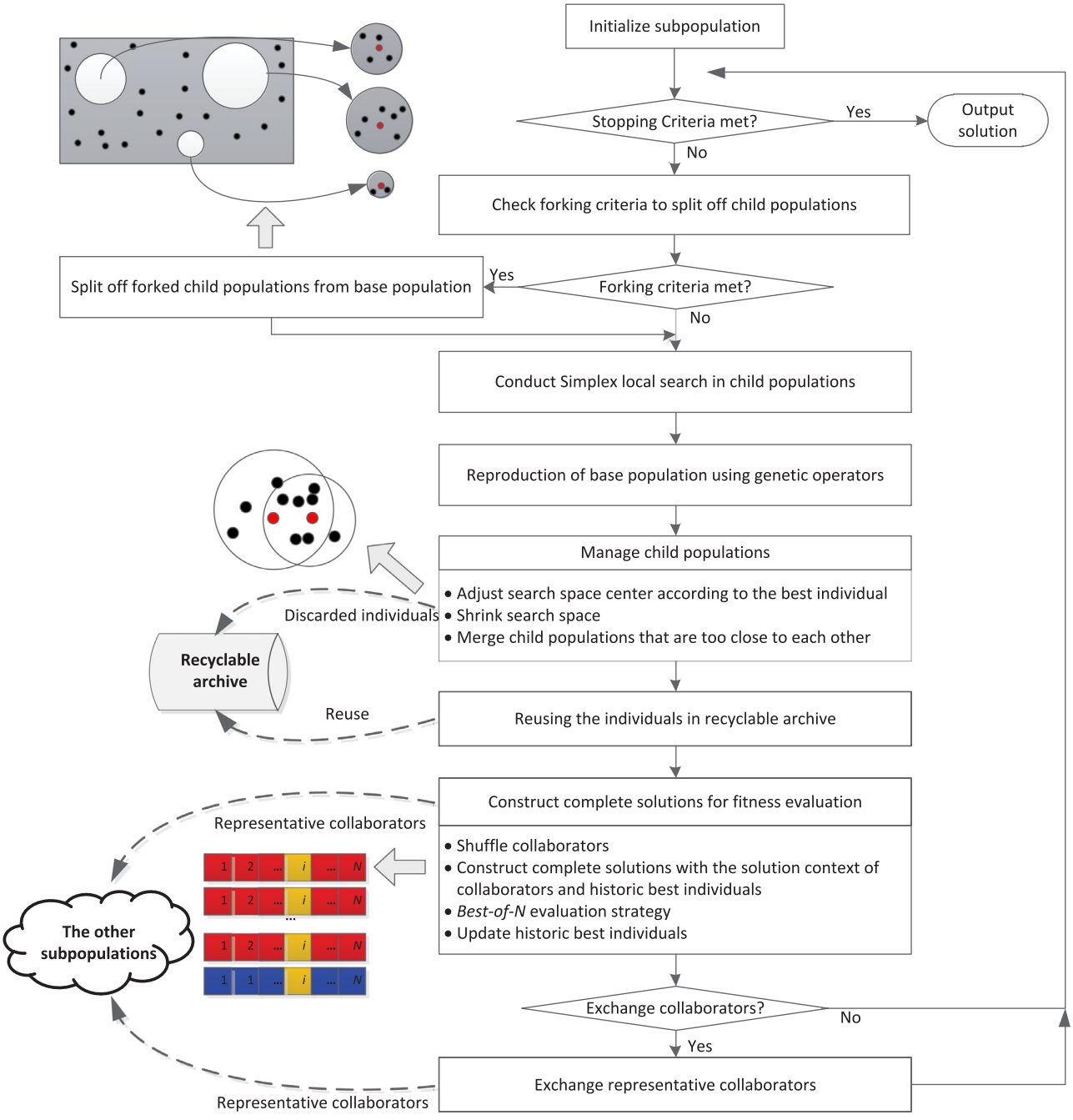


Fig. 5. Flowchart of each subpopulation.

0.75 of the total number of generations, at which point it stays at 0 until the end of the run.

4.2.3. Complete CCEA (cCCEA)

cCCEA was used as a peer algorithm in [19] as an extreme case of the tCCEA. When conducting fitness evaluation, a subpopulation accesses all individuals of the other subpopulations to obtain the best-of-all evaluation. Obviously, such a greedy exchanging strategy is impractical for distributed computing systems especially when the time or energy cost consumed by communication cannot be ignored. However, we still use cCCEA (a basic version of greedy interaction based CCEAs) as a peer algorithm since several effective CCEAs (e.g. pCCEA [16], iCCEA [20] and Pareto CCGA [17]) have been developed with the assumption that the subpopulations can exchange all their individuals with each other or run in a centralized manner.

4.2.4. Cooperative co-evolutionary differential evolution (CCDE)

CCDE uses SaNSDE [56] which is a variant of differential evolution as the component optimizer. Recently, the CCDE was used to efficiently solve large scale test functions up to 1000 dimensions after automatically decomposing the problems into components by a differential grouping technique [12]. Different from the other three peer algorithms, CCDE evolves in a sequential manner. The evolution of subpopulations iterates in turn based on the best individuals (collaborators) just got by the other subpopulations in last iteration.

Table 1
Converging rates and statistical significance comparison of mCCEA, bCCEA, cCCEA, tCCEA and CCDE on MTQ problems over 50 runs. " < " means statistically significant worse.

H_2	Alg.	Rate	vs mCCEA
70	mCCEA	98%	N/A
	bCCEA	72%	<
	cCCEA	64%	<
	tCCEA	2%	<
	CCDE	22%	<
150	mCCEA	98%	N/A
	bCCEA	78%	<
	cCCEA	52%	<
	tCCEA	8%	<
	CCDE	26%	<
300	mCCEA	100%	N/A
	bCCEA	98%	<
	cCCEA	64%	<
	tCCEA	4%	<
	CCDE	40%	<

4.3. General experimental parameter settings

In the following experiments, mCCEA, tCCEA, bCCEA and cCCEA are implemented based on GA toolbox [57] with almost the default settings: the population size of each subpopulation is 50, tournament selection size is 2, simulated-binary crossover and polynomial mutation rates are 0.9 and 0.1 respectively. The maximal generation number of a run of each algorithm is 1000. All algorithms are run 50 times independently. The convergence criterion used in forking check and local search is that best fitness of a population keeps the same for more than $N_f = 5$ generations.

The following parameters related to multi-population mechanism: the maximal, minimal and initial size of child populations (i.e. P_{smax}, $P_{smin}$and P_{sinit}) are 20, 4 and 10 respectively. The minimal size of the base population P_{0min} is 10. Shrinking factor f_r is 2.0. The minimal radius of child populations r_{min} is 0.1. A subcomponent sends $Ncom = 4$ collaborators at each time.

4.4. Comparison with peer algorithms

In this section, we firstly compare the performance of mCCEA with bCCEA, tCCEA, cCCEA and CCDE on RO-featured problems, i.e. MTQ problems with different settings and analyze the population dynamics. Then the algorithms are compared on ten continuous optimization benchmark problems $F_1 \sim F_{10}$ to validate the effectiveness of mCCEA more generally.

4.4.1. Tests on MTQ problems

This set of experiments test the performance of aforementioned algorithms on MTQ problems to compare their ability to avoid the RO problem. Here we compare the algorithms with the following settings: H_2 is set to 70, 150 and 300 respectively and S_2 is set to 1/64. Other parameters are the same as that in Section 4.1.1. Table 1 shows the statistical results over 50 runs including convergence rates, statistical significance for comparison of best solutions and average number of fitness evaluation. Since the results often do not have a normal distribution, we verify the statistical significance via Wilcoxon rank sum test. Note that a convergence is recognized only when the fitness difference between output solution and the global optimum is less than 0.1.

From the Table 1, it can be seen that the proposed mCCEA performs the best and significantly outperforms the peer algorithms in all cases. It has a higher rate of convergence. Thanks to the multi-population mechanism, mCCEA can find good representatives of each subcomponent in dynamic landscapes to enhance the coevolution effectively. The advantages of these representatives lie not only in the high fitness value (they are the local or global optima) but also the high diversity due to the spatial distribution of the optima.

In addition to the good representatives, the use of the best collaborator in the previous generations also affects the performance significantly. Among these algorithms, mCCEA and bCCEA use the best collaborator in the previous generations by maintaining historic best individual of child populations and partly biasing the search according to the historic best individual respectively. The other three algorithms (i.e. cCCEA, tCCEA and CCDE) only use the current best individuals as collaborators. As seen from Table 1, mCCEA and bCCEA has higher converging rates than the other algorithms although all of the algorithms utilize the same *best-of-N* collaboration strategy.

Fig. 6 show the population dynamics of mCCEA, bCCEA, cCCEA, tCCEA and CCDE on three MTQ problems (H_2 is set to 70, 150 and 300 respectively). It can be seen that in both decision space and objective space the mCCEA can gradually converge to the global optimum. Although bCCEA has the similar dynamics tendency it failed to fully converge to the global optimum. This verifies the effectiveness of multi-population mechanism which effectively compensates information and help subpopulations coevolve to the global optimum. Also seen from the dynamics of cCCEA, tCCEA and CCDE, their co-evolution almost fall into standstill without maintaining historic best individuals' information. This indicates that historic best individuals should be considered together with the collaborators to evaluate the fitness of individuals.

4.4.2. Tests on $F_1 \sim F_{10}$

In this set of experiments, a more wide range of comparison is made. mCCEA is compared with bCCEA, tCCEA and CCDE on 10 continuous optimization problems selected from the benchmark suite of CEC 2015 competition on 'learning-based real-parameter single objective optimization'. Note that cCCEA is not considered because it is an extreme case of tCCEA and has little potential to be applied. Each algorithm iterates 200 generations in each run. The statistical results over 50 independent runs are given in Table 2. As suggested in [58], we conduct two different statistical comparisons among these algorithms. Firstly, for each test function, mCCEA is compared with each of the other three algorithms with Wilcoxon signed rank test. It can be seen that the proposed mCCEA performs significantly better than the three compared algorithms on all test functions.

Secondly, the average Friedman ranks of each algorithm are given for $F_1 \sim F_{10}$. We can see that the proposed mCCEA again performs the best on all test functions. Although CCDE adopts a simple interaction strategy, i.e., exchanging the best individual among subcomponents, it ranks the second on most of the test functions ($F_1 \sim F_9$). This might be attributed to its powerful optimizer, i.e. the SaNSDE.

Another interesting finding is that according to an additional Wilcoxon signed rank comparison between bCCEA and tCCEA, bCCEA only achieves better performance over tCCEA on F_4, F_5, F_6, F_9 but loses on F_2, F_3, F_8. However, as shown in Table 1, bCCEA shows faster convergence than tCCEA on MTQ problems. This means that it is difficult for bCCEA to properly estimate the optimal collaborators when the landscape is complicated. In contrast, the information compensation of mCCEA works effectively for both simple and complex landscapes. As a result, mCCEA significantly outperforms the compared algorithms on all test functions (MTQ problems and $F_1 \sim F_{10}$).

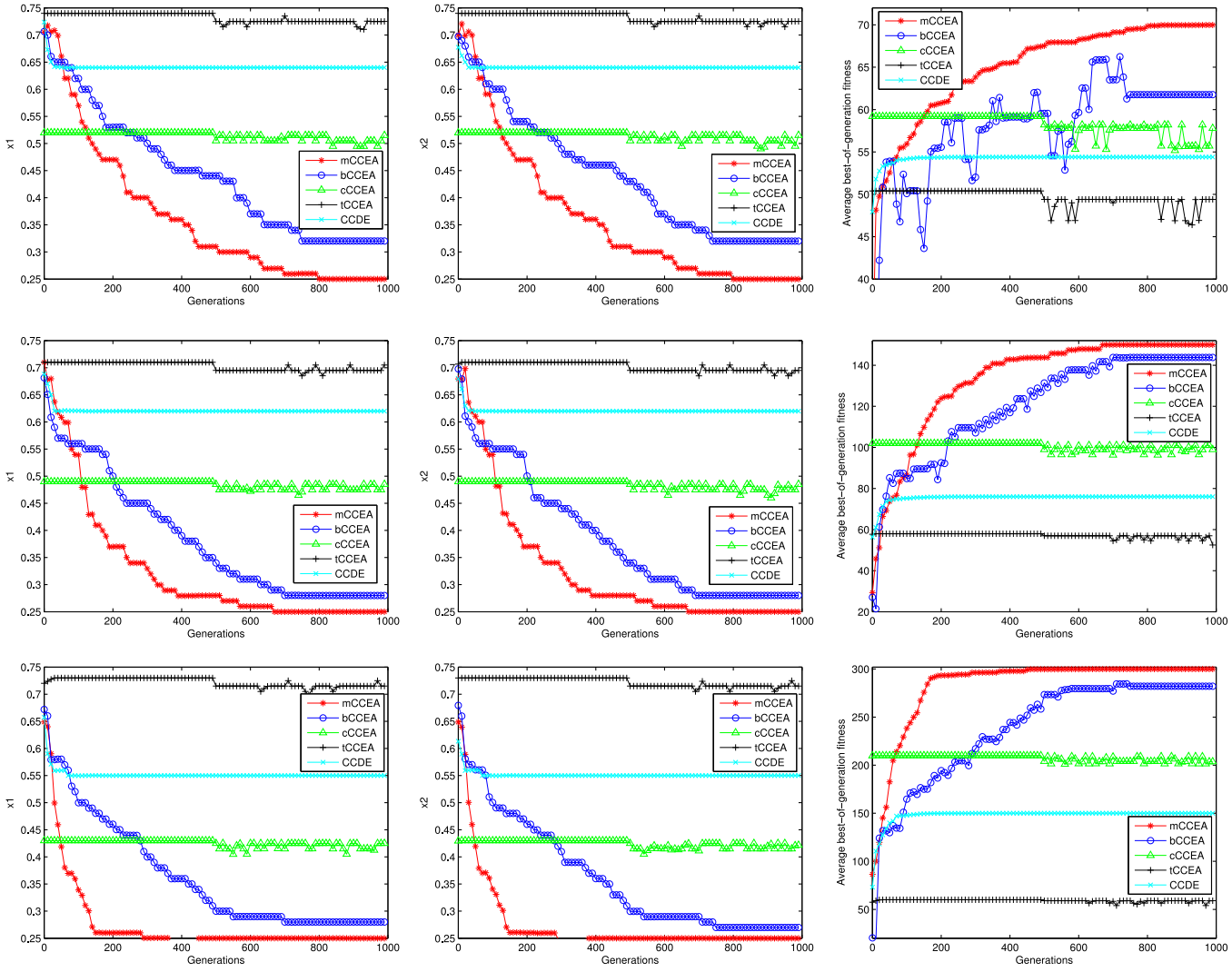


Fig. 6. Population dynamics (average values of x_1 (*left column*), x_2 (*middle column*) and best fitness (*right column*) against generations) of mCCEA, bCCEA, cCCEA, tCCEA and CCDE on MTQ problems with different configurations: $H_2 = 70$ (*top row*), $H_2 = 150$ (*middle row*), $H_2 = 300$ (*bottom row*).

4.5. Sensitivity analysis

In the following two sets of experiments, the effect of varying the collaborators exchanging interval and forking parameters used in the multi-population mechanism are tested and analyzed on MTQ problems.

4.5.1. Effect of varying the interaction interval

The objective of this set of experiments is to compare mCCEA with other algorithms regarding the potential of being applied to distributed computing systems. From the practical point of view, the frequency and amount of data exchanging are the most important factors of a distributed computing system. In this set of experiments the mCCEA, bCCEA, cCCEA, tCCEA and CCDE are compared with different collaborator exchanging intervals (every 1, 10 50 and 100 generations). Nine MTQ problems are used by setting H_2 to 70, 150, 300 and setting S_2 to 1/128, 1/64, 1/32 respectively.

Fig. 7 shows the performance comparison on MTQ problems with different collaborator exchanging intervals. It can be seen that the mCCEA again outperforms the other algorithms on MTQ problems with different collaborator exchanging intervals. This is because the multi-population mechanism provides the subpopulations with better representative collaborators. Besides, the maintaining of historical best individuals helps the subpopulation store the historic collaborative information and utilize it effectively. Both of these two benefits help subpopulations compensate information for effective co-evolution.

The sensitivity to exchanging interval can also be observed in Fig. 7. Considering the mCCEA and the bCCEA which perform better seen from the globally optimal convergence and population dynamics in the above experiments, their performance decreases when exchanging interval increases. This is because a large exchanging interval means less opportunity to update representative collaborators and historic best individuals. Both of them are the key factors for information compensation. In addition, comparing with the bCCEA, the mCCEA is more robust to exchanging interval. This can be explained from two perspectives. First, the bCCEA only maintains the historic best individual of a subpopulation while the mCCEA maintains the historic best individual of each child population; Second, in addition to multiple historic best individuals, the mCCEAs further maintains multiple populations which provide more diversity conduct co-evolution. Obviously, the mCCEA achieves more effective information compensation in both perspectives.

4.5.2. Effect of varying the forking parameters

As mentioned in Section 3.2.1, a forking process will be triggered when at least P_{sinit} individuals locate in the subspace

Table 2
Statistical performance comparison on minimization problems $F_1 \sim F_{10}$ over 50 runs. The values under function numbers are the fitness values of global optima. Pair-wise Wilcoxon signed ranks test results are given with a significance level of 0.05. Friedman ranks are averaged to conduct multiple comparisons (the smaller a rank is the better an algorithm performs).

Function	Algorithm	Best	worst	Median	Mean	Std	mCCEA vs. (Wilcoxon)	Friedman ranks
	mCCEA	100.00138	740.6148	169.8137	230.834	160.6283	/	38.1
F1	bCCEA	103.96952	9.13E+08	5389.922	43562100	1.55E+08	s better	142.82
(100)	tCCEA	215.56448	1.92E+08	3298.751	3888020	27099800	s better	133.98
	CCDE	101.3109	10391.47	922.6851	1772.84	2137.56	s better	87.1
	mCCEA	200.00193	811.4143	242.3695	293.768	136.6079	/	36.94
F2	bCCEA	239.38716	2.28E+09	5440.256	1.46E+08	4.76E+08	s better	146.54
(200)	tCCEA	212.27575	19181600	2499.051	425506.5	2719040	s better	128.54
	CCDE	201.7248	10568.17	688.7958	1883.252	2433.24	s better	89.98
	mCCEA	300.00019	300.0183	300.003	300.0042	0.00386	/	25.52
F3	bCCEA	300.06996	304.0405	302.106	302.0221	0.96074	s better	154.96
(300)	tCCEA	300.06883	304.5203	301.0908	301.4869	1.14052	s better	135.62
	CCDE	300.0182	301.5586	300.2489	300.3385	0.31951	s better	85.9
	mCCEA	400	400	400	400	5.16E-06	/	25.5
F4	bCCEA	400.00273	426.1962	401.0002	401.3158	3.64282	s better	125.55
(400)	tCCEA	400.00029	430.8183	401.0687	404.0526	7.74149	s better	150.65
	CCDE	400.0023	401.0743	400.9954	400.5716	0.49534	s better	100.3
	mCCEA	500	500.3124	500	500.0625	0.12613	/	30.7
F5	bCCEA	500.03048	932.5005	501.1119	514.9861	64.69657	s better	130.08
(500)	tCCEA	500.05101	1586.878	503.0184	675.822	332.3073	s better	146.78
	CCDE	500.0019	517.9151	500.404	502.142	5.12343	s better	94.44
	mCCEA	900.00016	1000.017	900.0025	903.006	15.67502	/	30.26
F6	bCCEA	900.02175	1001.049	902.5865	933.677	46.25903	s better	131.06
(900)	tCCEA	900.05655	1608.672	1000.075	980.4748	123.4655	s better	144.14
	CCDE	900.0077	907.3563	900.9119	901.2234	1.33708	s better	96.54
	mCCEA	1100.02128	1109.954	1100.19	1100.438	1.38351	/	27.7
F7	bCCEA	1101.58548	1208.006	1108.569	1110.83	15.47258	s better	133.82
(1100)	tCCEA	1102.06202	1199.35	1108.456	1113.123	18.68598	s better	142.54
	CCDE	1101.185	1160.954	1104.055	1105.668	8.38916	s better	97.94
	mCCEA	1200.00552	1308.645	1200.037	1216.3	37.65394	/	40.86
F8	bCCEA	1201.06302	1853.062	1352.446	1333.774	97.96627	s better	166.98
(1200)	tCCEA	1200.28921	1323.372	1208.174	1244.425	50.10516	s better	125.96
	CCDE	1200.072	1201.923	1200.472	1200.518	0.34991	s better	68.2
	mCCEA	1400.47889	1401.473	1401.058	1401.046	0.29018	/	25.5
F9	bCCEA	1406.0355	7796.524	1428.559	1589.633	897.9504	s better	120.98
(1400)	tCCEA	1402.75172	14191.9	1538.08	2959.408	3066.013	s better	149.3
	CCDE	1402.413	1711.535	1414.758	1470.322	84.29948	s better	106.22
	mCCEA	1500.04472	1600.022	1600.005	1591.199	27.72482	/	25.5
F10	bCCEA	1600.05782	1602.054	1600.48	1600.611	0.42575	s better	111.56
(1500)	tCCEA	1600.10496	4335.618	1600.532	1661.34	387.9274	s better	118.08
	CCDE	1600.199	1603.054	1600.943	1601.086	0.64886	s better	146.86

Table 3
Convergence rates of mCCEA with different P_{sinit} and r_{max} over 50 runs

P_{sinit}	$r_{max} = 0.15$	$r_{max} = 0.30$	$r_{max} = 0.45$
10	84%	72%	64%
20	26%	22%	14%
30	18%	10%	10%

determined by the best individual and the maximal radius of child population (r_{max}) subject to that the improvement of the best individual stay stationary for $N_f = 5$ generations. The objective of this set of experiments is to analyze to sensitivity of mCCEA to P_{sinit} and r_{max} with the settings of 10, 20, 30 and 0.15, 0.30, 0.45 respectively. Note that all the experiments here are based on MTQ problems with $H_2 = 70$ and $S_2 = 1/64$.

Table 3 shows the number of convergence when P_{sinit} and r_{max} are given different settings. Fig. 8 demonstrates the performance of mCCEA with different P_{sinit} and r_{max}. It can be seen that both of P_{sinit} and r_{max} affect the performance of mCCEA significantly. As P_{sinit} or r_{max} increases the performance of mCCEA decreases.

The reason of this tendency lies in the effects of these two parameters on the forking process. A smaller P_{sinit} means easier to match the forking criteria and more child populations could be generated subject to the limitation of individual resource. Thanks to this, more child populations can be easily split off to track the local or global optima, which is good for collect representative collaborators for co-evolution.

It is interesting that a lager r_{max} will lead to worse performance. Although a larger r_{max} also means easier to trigger forking process, it limits the number of existing child populations because a larger r_{max} makes the search spaces of child populations easier to overlap and child populations are like to merge. This may lead to insufficient number of child populations to concurrently search multiple optima, which influences the information compensation.

The above statement can also be verified by Fig. 9 which shows the average number of child populations at each generation. It can be seen that more child populations are generated when P_{sinit} is set to 10 and the number of child populations gradually adjusts to about 4. Almost each subpopulation maintains 2 child populations to track the local or global optimum, which is a proper reaction to the global landscapes. Also, the decrease of the number of child

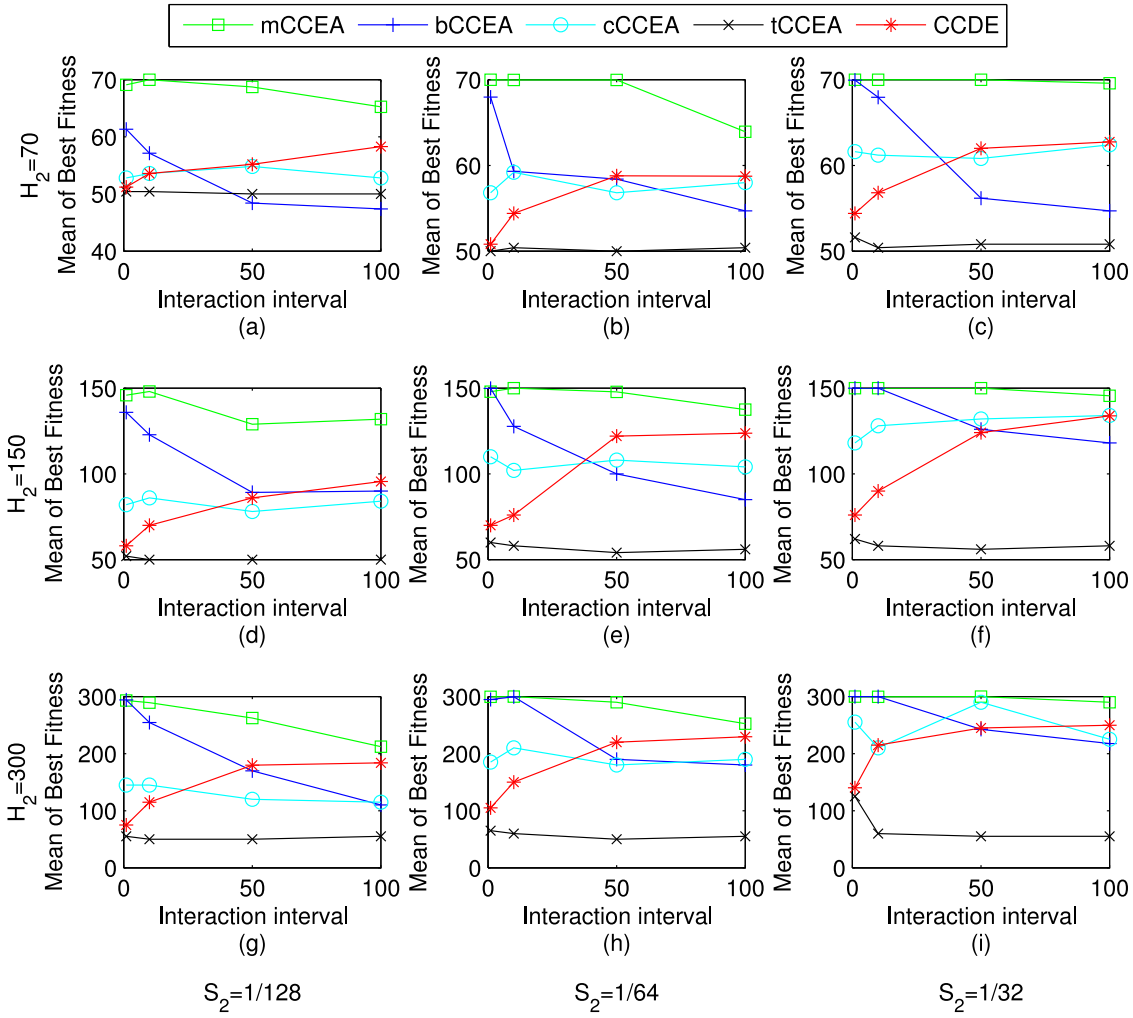


Fig. 7. Performance comparison of mCCEA, bCCEA, cCCEA, tCCEA and CCDE with different interaction intervals on MTQ problems.

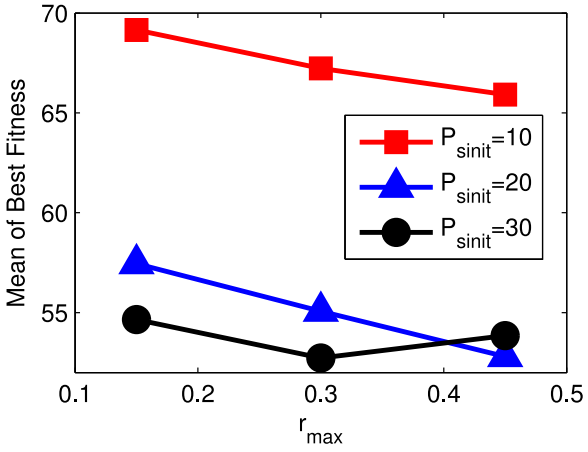


Fig. 8. Performance of mCCEA with different P_{sinit} and r_{max} averaged over 50 runs.

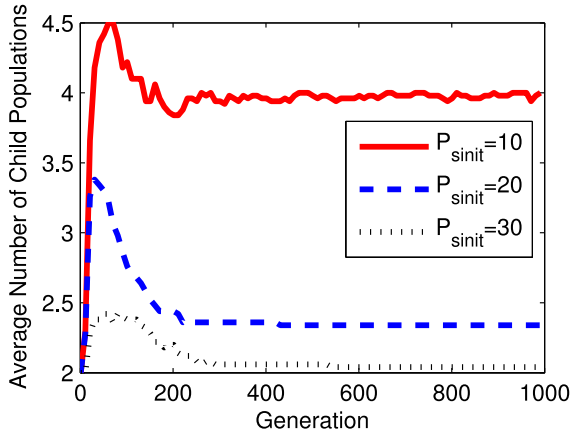
populations can be observed in Fig. 9 (a) (b) and (c) when r_{max} increases.

5. Conclusions and future work

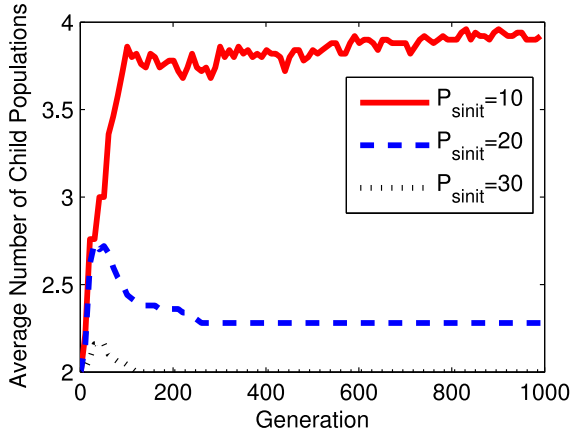
In order to overcome the inherent problems of CCEAs like the RO problem, a multi-population mechanism base CCEA (mCCEA) has been proposed considering the fact that the landscapes of the subcomponents are dynamic determined by the interactive nature of CCEAs. In the mCCEA, a subpopulation is composed of a base and several child populations. These populations conduct dynamic multimodal optimization so as to obtain local and global optima as the representative collaborators with high fitness and diversity. The simplex local search is introduced into child populations for efficiently searching in their relative small search space. To better maintain and utilize the collaborators, the historic best individuals of the base and child populations are recorded along with their complete solution context, which is beneficial for maintaining more useful co-evolutionary information and simplifying the forking process in the multi-population mechanism.

In order to justify the proposed mCCEA, experiments have been carried out to compare mCCEA with several peer CCEAs on a set of MTQ problems with different difficulty and ten benchmark problems on continues optimization used in 'CEC'15 competition on learning-based real-parameter single objective optimization'. In addition, the sensitivity analysis was also done with regarding the key parameters involved in the mCCEA.

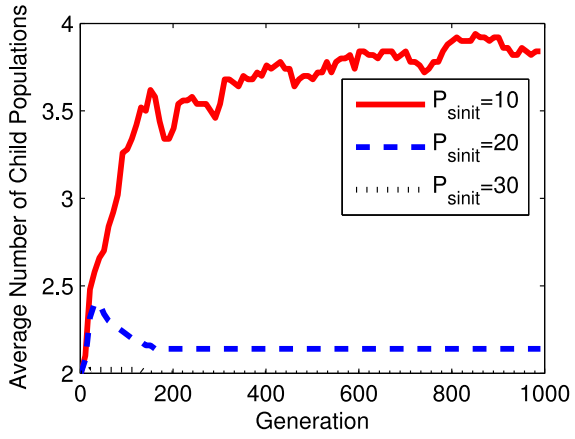
From the experimental results the following conclusions can be drawn. The proposed mCCEA performs the best and significantly outperforms the peer algorithms in all RO-featured test problems (i.e. MTQ problems). It has a higher convergence rate to global optimum but with less fitness evaluations. The proposed mCCEA not only performs better but also shows practical feature for



(a) $r_{max} = 0.15$.



(b) $r_{max} = 0.30$.



(c) $r_{max} = 0.45$.

Fig. 9. Average number of child populations against generations through out 50 runs.

distributed computing. Besides, better performance and robust optimization can also be observed even when collaborator exchanging interval is large. Moreover, ten non-separable functions are also used to test and compare the algorithms. The statistical results show a wide range of verification of the effectiveness and advantage over the peer algorithms. The sensitivity analysis show that improperly large values of initial size and maximal radius of child population may make the forking process more difficult to be triggered and limit the number of child populations, which will negatively affect the performance of mCCEA.

This work demonstrates that the idea of adopting a dynamic optimization approach to achieve information compensation for CCEAs is promising. Nevertheless, in the present work, the interaction interval is fixed and cannot adapt to the co-evolutionary process. Actually, as suggested in [59], more collaborators should be used in early generations so that the algorithm can benefit from a varying set of collaborators. On the contrary, only a smaller number of collaborators should be used in later generations. Therefore, our future work will focus on how to adaptively trigger the interaction. In addition, more effort should be made to apply the proposed work to solving real-world problems, such as complex recommend systems [60].

Acknowledgments

This work has been supported by National Nature Science Foundation of China (No. 61105068 and 61473233), Fundamental Research Funds for the Central Universities of China under Grant 3102016ZY007, an EPSRC grant (No. EP/M017869/1), and the Joint Research Fund for Overseas Chinese, Hong Kong and Macao Scholars of the National Natural Science Foundation of China (No. 61428302).

References

[1] M.A. Potter, The design and analysis of a computational model of cooperative coevolution, 1997 Ph.D. thesis.
[2] M.A. Potter, K.A. De Jong, Cooperative coevolution: an architecture for evolving coadapted subcomponents, Evol. Comput. 8 (1) (2000) 1–29.
[3] A. Carvalho, A cooperative coevolutionary genetic algorithm for learning Bayesian network structures, in: 13th Annual Conference on Genetic and Evolutionary Computation, ACM, 2011, pp. 1131–1138. 2001729
[4] W. Ding, J. Wang, A novel approach to minimum attribute reduction based on quantum-inspired self-adaptive cooperative co-evolution, Know. Based Syst. 50 (2013) 1–13.
[5] K.C. Tan, Y.J. Yang, C.K. Goh, A distributed cooperative coevolutionary algorithm for multiobjective optimization, IEEE Trans. Evol. Comput. 10 (5) (2006) 527–549.
[6] A.S. Ruela, L.L. Aquino, F.G. Guimares, A cooperative coevolutionary algorithm for the design of wireless sensor networks, in: 13th Annual Conference Companion on Genetic and Evolutionary Computation, ACM, 2011, pp. 607–614. 2002056
[7] M. Colby, M. Knudson, K. Tumer, Multiagent flight control in dynamic environments with cooperative coevolutionary algorithms, In Association for the Advancement of Artificial Intelligence, Modeling in Human Machine Systems: Challenges for Formal Verification 2014, AAAI, 2014.
[8] M.A. Potter, C. Couldrey, A cooperative coevolutionary approach to partitional clustering, in: 11th International Conference on Parallel Problem Solving from Nature, vol. 1, Springer-Verlag, 2010, pp. 374–383. 1885072
[9] Z. Yang, K. Tang, X. Yao, Multilevel cooperative coevolution for large scale optimization, in: Proceedings of the 2008 IEEE Congress on Evolutionary Computation, 2008, pp. 1663–1670.
[10] W. Chen, T. Weise, Z. Yang, K. Tang, Large-scale global optimization using cooperative coevolution with variable interaction learning, in: Proceedings of the 11th International Conference on Parallel Problem Solving from Nature, vol. 6239, LNCS, 2010, pp. 300–309.
[11] X. Li, Y. Xin, Cooperatively coevolving particle swarms for large scale optimization, IEEE Trans. Evol. Comput. 16 (2) (2012) 210–224.
[12] M.N. Omidvar, X. Li, Y. Mei, X. Yao, Cooperative co-evolution with differential grouping for large scale optimization, IEEE Trans. Evol. Comput. 18 (3) (2014) 378–393.
[13] R.P. Wiegand, An analysis of cooperative coevolutionary algorithms, 2003 Ph.D. thesis.
[14] S.G. Ficici, J.B. Pollack, A game-theoretic memory mechanism for coevolution, in: Proceedings of the 2003 Genetic and Evolutionary Computation Conference, 2003, pp. 286–297.
[15] E.P. Manning, Coevolution in a large search space using resource-limited nash memory, in: Proceedings of the 2010 Genetic and Evolutionary Computation Conference, ACM, 2010, pp. 999–1006.
[16] A. Bucci, J.B. Pollack, On identifying global optima in cooperative coevolution, in: Proceedings of the 2005 Genetic and Evolutionary Computation Conference, 2005, pp. 539–544.
[17] M. Shi, H. Wu, Pareto cooperative coevolutionary genetic algorithm using reference sharing collaboration, in: Proceedings of the 2009 Genetic and Evolutionary Computation Conference, ACM, 2009, pp. 867–874.
[18] L. Panait, S. Luke, R.P. Wiegand, Biasing coevolutionary search for optimal multiagent behaviors, IEEE Trans. Evol. Comput. 10 (6) (2006) 629–645.

[19] L. Panait, S. Luke, Selecting informative actions improves cooperative multiagent learning, in: 5th International Joint Conference on Autonomous Agents and Multiagent Systems, 2006, pp. 760–766.
[20] L. Panait, S. Luke, J.F. Harrison, Archive-based cooperative coevolutionary algorithms, in: Proceedings of the 2006 Genetic and Evolutionary Computation Conference, 2006, pp. 345–352.
[21] J. Branke, Evolutionary Optimization in Dynamic Environments, Kluwer Academic Pub, 2002.
[22] Y. Jin, J. Branke, Evolutionary optimization in uncertain environments a survey, IEEE Trans. Evol. Comput. 9 (3) (2005) 303–317.
[23] T.T. Nguyen, S. Yang, J. Branke, Evolutionary dynamic optimization: a survey of the state of the art, Swarm Evol. Comput. 6 (October) (2012) 1–24.
[24] S. Yang, X. Yao, Evolutionary Computation for Dynamic Optimization Problems, Studies in Computational Intelligence, Springer-Verlag Berlin Heidelberg, 2013.
[25] X. Peng, X. Lei, K. Liu, Compensate information from multimodal dynamic landscapes: an anti-pathology cooperative coevolutionary algorithm, in: Proceedings of the 2014 IEEE Congress on Evolutionary Computation, 2014, pp. 2578–2584.
[26] H.G. Cobb, An Investigation into the Use of Hypermutation as an Adaptive Operator in Genetic Algorithms Having Continuous, Time-Dependent Nonstationary Environments, Technical Report AIC-90-001, Naval Res. Lab, 1990.
[27] F. Vavak, K. Jukes, T.C. Fogarty, Performance of a genetic algorithm with variable local search range relative to frequency of the environmental changes, in: In Genetic Programming 1998: Proceedings of the Third Annual Conference, Morgan Kaufmann, 1998, pp. 22–25.
[28] Y. Woldesenbet, G. Yen, Dynamic evolutionary algorithm with variable relocation, IEEE Trans. Evol. Comput. 13 (3) (2009) 500–513.
[29] J.J. Grefenstette, J. Fitzpatrick, Genetic algorithms for changing environments, in: R. Maenner, B. Manderick (Eds.), the 2nd International Conference on Parallel Problem Solving from Nature (PPSN II), 1992, pp. 137–144.
[30] R.W. Morrison, Designing Evolutionary Algorithms for Dynamic Environments, Springer, 2004.
[31] S. Yang, Experimental study on population-based incremental learning algorithms for dynamic optimization problems, Soft Comput. 9 (11) (2005) 815–834.
[32] L.T. Bui, J. Branke, H.A. Abbass, Multiobjective Optimization for Dynamic Environments, Technical Report, The Artificial Life and Adaptive Robotics Laboratory, School of Information Technology and Electrical Engineering, University of New South Wales, 2005.
[33] J. Lewis, E. Hart, G. Ritchie, A comparison of dominance mechanisms and simple mutation on non-stationary problems, in: A.E. Eiben, T. Back, M. Schoenauer, H.P. Schwefel (Eds.), Parallel Problem Solving from Nature, LNCS1498, Springer, 1998, pp. 139–148.
[34] K.P. Ng, K.C. Wong, A new diploid scheme and dominance change mechanism for non-stationary function optimization, in: 6th International Conference on Genetic Algorithms, Morgan Kaufmann, 1995, pp. 159–166.
[35] A.i. Uyar, A.E. Harmanci, A new population based adaptive domination change mechanism for diploid genetic algorithms in dynamic environments, Soft Comput. 9 (11) (2005) 803–814.
[36] J. Branke, Memory enhanced evolutionary algorithms for changing optimization problems, in: Proceedings of the 1999 IEEE Congress on Evolutionary Computation, vol. 3, IEEE, 1999, pp. 1875–1882.
[37] S. Yang, X. Yao, Population-based incremental learning with associative memory for dynamic environments, IEEE Trans. Evol. Comput. 12 (5) (2008) 542–561.
[38] M. Daneshyari, G. Yen, Dynamic optimization using cultural based pso, in: Proceedings of the 2011 IEEE Conference on Evolutionary Computation, 2011, pp. 509–516.
[39] A. Simos, E. Costa, Memory-based chc algorithms for the dynamic traveling salesman problem, in: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, ACM, 2011, pp. 1037–1044.
[40] X. Peng, X. Gao, S. Yang, Environment identification based memory scheme for estimation of distribution algorithms in dynamic environments, Soft Comput. 15 (2) (2011) 311–326.
[41] C. Rossi, M. Abderrahim, J.C. Diaz, Tracking moving optima using kalman-based predictions, Evol. Comput. 16 (1) (2008) 1–30.
[42] A. Simoes, E. Costa, Prediction in evolutionary algorithms for dynamic environments using markov chains and nonlinear regression, in: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, ACM, 2009, pp. 883–890.
[43] A. Zhou, Y. Jin, Q. Zhang, A population prediction strategy for evolutionary dynamic multiobjective optimization, IEEE Trans. Cybern. 44 (1) (2014) 40–53.
[44] F. Oppacher, M. Wineberg, The shifting balance genetic algorithm: Improving the ga in a dynamic environment, in: W. Banzhalf (Ed.), Genetic and Evolutionary Computation Conference, vol. 1, Morgan Kaufmann, 1999, pp. 504–510.
[45] R.K. Ursem, Multinational ga optimization techniques in dynamic environments, in: D.W.e. al. (Ed.), Proceedings of the 2nd annual conference on genetic and evolutionary computation conference, Morgan Kaufmann Publishers, 2000, pp. 19–26.
[46] T. Blackwell, J. Branke, Multiswarms, exclusion, and anti-convergence in dynamic environments, IEEE Trans. Evol. Comput. 10 (4) (2006) 459–472.
[47] D. Parrott, X. Li, Locating and tracking multiple dynamic optima by a particle swarm model using speciation, IEEE Trans. Evol. Comput. 10 (4) (2006) 440–458.
[48] S. Yang, C. Li, A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments, IEEE Trans. Evol. Comput. 14 (6) (2010) 959–974.
[49] S.K. Nseef, S. Abdullah, A. Turky, G. Kendall, An adaptive multi-population artificial bee colony algorithm for dynamic optimisation problems, Knowl. Based Syst. 104 (2016) 14–23.
[50] R. Mendes, A. Mohais, Dynde: a differential evolution for dynamic optimization problems, in: Proceedings of the 2005 IEEE Conference on Evolutionary Computation, IEEE, 2005.
[51] S. Tsutsui, Y. Fujimoto, Forking genetic algorithms: gas with search space division schemes, Evol. Comput. 5 (1) (1997) 61–79.
[52] J.A. Nelder, R. Mead, A simplex method for function minimization, Comput. J. 7 (1965) 308–313.
[53] L. Panait, Theoretical convergence guarantees for cooperative coevolutionary algorithms, Evol. Comput. 18 (4) (2010) 581–615.
[54] J.J. Liang, B.Y. Qu, P.N. Suganthan, Q. Chen, Problem Definitions and Evaluation Criteria for the CEC 2015 Competition on Learning-based Real-Parameter Single Objective Optimization, Technical Report 201411A, Computational Intelligence Laboratory, Zhengzhou University and Nanyang Technological University, 2014.
[55] M.A. Potter, K.A. De Jong, A cooperative coevolutionary approach to function optimization, in: Parallel Problem Solving from Nature (PPSN III), Springer, 1994, pp. 249–257.
[56] Z. Yang, K. Tang, X. Yao, Self-adaptive differential evolution with neighborhood search, in: IEEE Congress on Evolution Computation, IEEE, 2008, pp. 1110–1116.
[57] K. Sastry, Single and Multiobjective Genetic Algorithm Toolbox in C++, Technical Report, IlliGAL, 2007.
[58] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm Evol. Comput. 1 (1) (2011) 3–18.
[59] L. Panait, S. Luke, Time-dependent collaboration schemes for cooperative coevolutionary algorithms, in: Proceedings of the 2005 AAAI Fall Symposium on Coevolutionary and Coadaptive Systems, 2005.
[60] S. Wang, M. Gong, H. Li, J. Yang, Multi-objective optimization for long tail recommendation, Knowl. Based Syst. 104 (2016) 145–155.