An Intelligent Multi-Restart Memetic Algorithm for Box Constrained Global Optimisation

J. Sun*
J.Sun@abertay.ac.uk
CPIB, School of Bioscience, The University of Nottingham,
Sutton Bonington, LE12 5RD, United Kingdom

J. M. Garibaldi
jmg@cs.nott.ac.uk
Department of Computer Science, The University of Nottingham,
Nottingham, NG8 1BB, United Kingdom

N. Krasnogor
nxk@cs.nott.ac.uk
Department of Computer Science, The University of Nottingham,
Nottingham, NG8 1BB, United Kingdom

Q. Zhang
qzhang@essex.ac.uk
Department of Computing and Electrical Systems, The University of Essex,
Colchester, CO4 3SQ, United Kingdom

Abstract

In this paper, we propose a multi-restart memetic algorithm framework for box constrained global continuous optimisation. In this framework, an evolutionary algorithm (EA) and a local optimizer are employed as separated building blocks. The EA is used to explore the search space for very promising solutions (e.g., solutions in the attraction basin of the global optimum) through its exploration capability and previous EA search history, and local search is used to improve these promising solutions to local optima. An estimation of distribution algorithm (EDA) combined with a derivative free local optimizer, called NEWUOA (M. Powell, Developments of NEWUOA for minimization without derivatives. *Journal of Numerical Analysis*, 28:649–664, 2008), is developed based on this framework and empirically compared with several well-known EAs on a set of 40 commonly used test functions. The main components of the specific algorithm include: (1) an adaptive multivariate probability model, (2) a multiple sampling strategy, (3) decoupling of the hybridisation strategy, and (4) a restart mechanism. The adaptive multivariate probability model and multiple sampling strategy are designed to enhance the exploration capability. The restart mechanism attempts to make the search escape from local optima, resorting to previous search history. Comparison results show that the algorithm is comparable with the best known EAs, including the winner of the 2005 IEEE Congress on Evolutionary Computation (CEC2005), and significantly better than the others in terms of both the solution quality and computational cost.

Keywords

Global numerical optimisation, estimation of distribution algorithm, local search, memetic algorithm.

*Corresponding Author.

1 Introduction

In this paper, we consider the following box constrained optimisation problem:

$$\min_{\mathbf{x}=(x_1,\cdots,x_n)\in\mathcal{D}\subset\mathbb{R}^n} f(\mathbf{x}) \tag{1}$$

where $\mathcal{D}$ is the search space with $x_i \in [a_i, b_i]$, $1 \leq i \leq n$ and n is the dimensionality of the optimisation problem. From the 1970s, population-based modern global optimisation methods, such as evolutionary programming (Fogel, 1999; Yao and Liu, 1999; Lee and Yao, 2004), genetic algorithms (Michalewicz, 1996; Tsai et al., 2004; Tu and Lu, 2004; Wang and Dang, 2007; Leung and Wang, 2001; Zhong et al., 2004), estimation of distribution algorithms (Larrañaga and Lozano, 2002; Zhang et al., 2003), and many others have been proposed and achieved great success in theory and practice.

One of the main concerns in the design of evolutionary algorithms (EAs) is to balance the exploration and exploitation aspects (Bäck, 1996; Bäck et al., 1997; Torn and Zilinskas, 1989) for effective and efficient search. Among these EAs, memetic algorithms (MAs), that is, the combination of local search (LS) and EAs (Moscato, 1989), have been considered as a promising paradigm. MAs aim to improve the slow convergence of EAs to locate high-quality solutions by using LS to exploit the local fitness landscape. Readers are referred to Hart et al. (2005) for recent advances in MAs, and Krasnogor and Smith (2000) or Krasnogor and Smith (2005) for a comprehensive review of MAs in solving optimisation problems. Also, see Krasnogor (2009) for a more detailed discussion of the balance between exploration and exploitation in MAs.

To design an efficient MA, the trade-off between the intensities of EA and LS methods needs to be carefully tuned under a fixed computational budget. Various strategies have been proposed in the literature to address the trade-off, particularly for continuous optimisation. As summarised elsewhere (Nguyen, Ong, and Krasnogor, 2007; Nguyen et al., 2009), the basic issues addressed in these strategies include: (i) the LS frequency, or similarly, the LS probability, (i.e., how often the LS is applied); (ii) the LS intensity (i.e., for how long to apply the LS); (iii) the subset of individuals selected to carry out LS; and (iv) the selection of memes (i.e., local optimizers) to be used. The study in Nguyen, Ong, and Krasnogor (2007) showed that the LS frequency and intensity should be balanced under a fixed computational budget, and it is better to apply the LS to the best individuals. Theoretical analysis of convergence and search speed of MAs can be found in Sudholt (2006, 2009). Extensive studies on the selection of memes, such as adaptive MAs (Ong and Keane, 2004; Ong et al., 2006; Houck et al., 2004) and co-evolving MAs (Smith, 2003, 2007), have shown that the memes employed have a major influence on the search performance of the MAs. Interested readers are referred to Ong et al. (2006) and the references therein. In this paper, we focus on the development of a MA with a single LS.

Hart (1994) proposed several mechanisms to decide which solutions LS should be applied to, and the LS frequency, including a fixed frequency method, a fitness-based adaptive method, and a distribution-based adaptive method. In the fixed frequency method, LS is to be used in a fixed frequency of generations to all new offspring. The fitness-based adaptive method biases the application of LS to promising solutions with an LS probability. The distribution-based adaptive method applies LS only to solutions that are far away from each other, or without redundancy in the population.

In Bambha et al. (2004), simulated heating is applied to vary an LS parameter (which specifies the LS intensity) during the optimisation process. In Molina et al. (2005), the LS intensity and probability are decided according to the individual qualities in the

population at each generation. An MA based on LS chains is developed in Molina et al. (2010) where previous LS parameters are inherited for the application of LS to newly generated individuals with fixed LS intensity, and the percentage of evaluations spent on LS is fixed. In Nguyen, Ong, Lim, et al. (2007), an adaptive method has been proposed for the selection of proper individuals to undertake LS. The probabilistic memetic framework developed in Nguyen et al. (2009) estimates the LS intensity for each individual at each generation. In Zhang et al. (2003), cheap and expensive LS algorithms are applied intelligently to solutions in different search stages. Cheap LS is applied to all new solutions to improve them within a limited number of steps. Expensive LS is only applied to the best solutions in every generation. The cheap LS can improve exploration ability, while the expensive LS will finally locate a near-global optimum.

Usually, for continuous optimisation problems, the memes used are classical LS algorithms, such as those methods appearing in the Schwefel libraries (Schwefel, 1995). Researchers have also proposed to use deliberately designed crossover operators or even an EA to take the role of LS in continuous domain optimisation, such as XLS (Satoh et al., 1996), SPX (Tsutsui et al., 1999; Tsutsui and Goldberg, 2002; Noman and Iba, 2008), and crossover LS (Jones, 1995; O'Reilly and Oppacher, 1995; Satoh et al., 1996; Lozano et al., 2004), among otheres. In Molina et al. (2010), CMA-ES (Hansen and Ostermeier, 2001) is applied as the LS algorithm. Although these methods may perform well, it is doubtful whether these heuristic-based local optimizers are more effective and efficient than classical LS algorithms for local improvement.

Informally, an LS is coupled with an EA in any existing MA. Usually, the LS takes effect on selected individuals at frequent generations (sometimes every generation). This could lead (in some cases) to the MA placing too much emphasis on exploitation. In other words, the balance of exploration and exploitation may be shifted too much in favour of exploitation (Houck et al., 1997). To redress the balance, we believe that it may sometimes be preferable to reduce the relative contribution of LS within the MA. Moreover, the computational cost to add a new solution to the overall MA process by EA and LS may also be highly unbalanced. It is rather easy for EA reproduction operators to generate such a solution, while LS usually requires a higher cost (although the cost does vary considerably depending on the LS intensity). In existing MAs, the imbalance is redressed by tuning the LS frequency and intensity. This tuning could make the development of an efficient MA rather complex.

In this paper, we attempt to moderate these problems in existing MAs. To realize this goal, we propose to isolate LS from the evolutionary search but not vice versa. That is, LS will be applied to improve a limited number of very promising solutions found by an EA in full strength, not to all or part of the population as in previous hybridisation strategies. Hence, we do not need to explicitly tune the LS frequency and intensity. The underlying idea of the algorithm design is to clearly distinguish exploration and exploitation. The task of the EA is limited to exploring the search space for very promising solutions, which is its advantage and should be strengthened. The task of the LS is to improve the very promising solutions efficiently to local optima. The less efforts contributed to the exploration for very promising solutions, the better. The design issues that arise are how to design an EA with good exploration capability and how to speed up an EA's exploration under a fixed computational budget. These problems are addressed in this paper.

The proposed algorithmic framework is described and discussed in Section 2. Section 3 describes the components of the exemplar algorithm including the adaptive

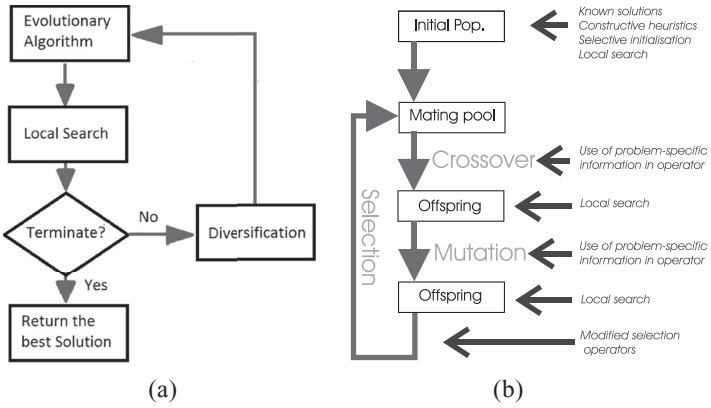


Figure 1: (a) The flowchart of the proposed novel framework and (b) the traditional MA framework as described in Eiben and Smith (2003).

multivariate model, multiple sampling strategy, and the embedding of these components. Section 4 gives the experimental results on a diverse set of commonly used test problems including those in the 2005 IEEE Congress on Evolutionary Computation (CEC2005; Suganthan et al., 2005) and the comparison with some well-known EAs. The effects of the algorithmic components, including the adaptive multivariate model, the multiple sampling strategy, the selection operation, the population size, and the decoupling hybridisation strategy, on the performance of the developed algorithm, and the scalability property of the developed algorithm, are experimentally studied in this section. Important issues related to the algorithmic framework and the developed algorithm are discussed in Section 5. Section 6 concludes the paper.

2 The Algorithmic Framework

Recall that one of the main merits of EAs is that they usually show good ability to explore the search space, while LSs are good at locating local optima. It is worth noting that the quality of the local optima found by LSs largely depends on the initial solutions they start from (although the parameters of the LS are also important, but basically it is the location of the initial solutions that decide the final solution quality). Imagine that if the exploration ability of an EA is extensively increased so that it can provide a very promising solution (e.g., a solution located in the attraction basin of a high-quality optimum, or the global optimum) in a short time, then a properly selected LS will improve the solution to the high-quality local optimum, or the global optimum, efficiently. If this idea can be realised, an efficient optimisation algorithm will be achieved.

The key design issues of an MA based on the above idea lies in the success of designing an EA that is able to quickly locate the very promising solution and the selection of a proper LS. However, it is not always guaranteed that an EA can find the near global optimum solution. Also, the EA's low exploration speed is not acceptable under a fixed computational budget.

In light of the above arguments, we propose a novel memetic framework, called intelligent multi-restart memetic algorithm (IMMA). Figure 1(a) displays the flowchart of the proposed framework. In the framework, LS and EA are untangled and committed to acting separately. That is, no LS is applied inside EA, which distinguishes the classical

Algorithm 1 The abstract IMMA algorithm framework

Input: The population size M, the size of the selected population K, and the restart criterion $\mathcal{C}$
Output: The best solution found $\mathbf{x}^*$

1. Initialization
2. Set $t := 0$, $c = 0$, $restart := 0$. Set $\mathbf{x}_c$, $\mathbf{x}^*$ and $\mathbf{x}_c^*$ as empty;
3. Diversification
4. **if** $restart = 1$, set $c := c + 1$ and $restart := 0$.
5. Randomly or heuristically generate a set of solutions as the initial population $\mathcal{P}(t)$. Evaluate $\mathcal{P}(t)$. Set $\mathbf{x}_c$ as the best solution in $\mathcal{P}(t)$.
6. Selection
7. Select K solutions from $\mathcal{P}(t)$ by a selection method;
8. Reproduction
9. Create a set of offspring according to the K selected solutions;
10. Replacement
11. Replace partially or fully $\mathcal{P}(t)$ by the created solutions;
12. Best Solution Updating
13. **if** the best solution in $\mathcal{P}(t+1)$ has a better fitness than $\mathbf{x}_c$, replace $\mathbf{x}_c$ with the best solution; Set $t := t + 1$;
14. Check Search Status
15. **if** the restart criterion $\mathcal{C}$ has been satisfied,
16. **then** perform LS taking $\mathbf{x}_c$ as the initial point. Set the resultant local optimum as $\mathbf{x}_c^*$.
17. Set $\mathbf{x}^*$ as the solution with the minimum objective function value among the $(c+1)$ solutions, and $restart := 1$.
18. **else** goto line 6.
19. Stop Criterion Check
20. **if** the stop criterion has not been met, goto line 3.
21. **else** Stop and return $\mathbf{x}^*$.

MA framework shown in Figure 1(b) where local search intertwines with EA operation. In the developed framework, local search comes into play only if an EA is considered to have found a very promising solution. The framework requires an EA with strong exploration ability and a fast exploration speed. If the termination criterion (e.g., fixed computational budget) has not been reached, the diversification scheme is applied so that the EA may escape from the local optima. This multi-restart strategy is used to increase the possibility of finding near global optimum solutions as opposed to the stochastic nature of EAs. Moreover, to enhance the efficiency of the new evolutionary search, it will be particularly useful if we can use knowledge obtained from previous visited solutions. The abstract algorithmic framework shown in Algorithm 1 describes the idea in more detail.

In the algorithmic framework, the loop from Selection to Check Search Status until convergence is called a cycle, that is, a typical run of an EA. The EA is considered to have converged if the restart criterion $\mathcal{C}$ is satisfied. The restart criterion $\mathcal{C}$ stops the EA's process of exploring for attraction basins of high quality solutions. After a cycle ends, local search is applied to improve the best solution found in the cycle. In Diversification, a new cycle begins following a finished cycle. We use $\mathbf{x}_c$ to record the best solution found in each cycle (called the cycle best), $\mathbf{x}_c^*$ is the local optimum resulting from the application of LS on $\mathbf{x}_c$, and $\mathbf{x}^*$ is the best solution among these local optima (called the global best).

In each cycle, the EA components can be specified at will, although the critical design issue is how to increase the exploration ability so that a promising solution can be found quickly. The restart criterion has an important role in deciding when a very promising solution is found. In our implementation, we consider the EA has found a possible very promising solution if in T consecutive generations, $\mathbf{x}_c$ is not updated. This does not guarantee that the current $\mathbf{x}_c$ is truly very promising; it may just be that the evolutionary search has been trapped in a large attraction basin. To tell exactly whether $\mathbf{x}_c$ is promising or not depends on knowledge about the fitness landscape of the optimisation problem. Unfortunately, this is not usually available. However, to choose between two solutions for the LS application, a direct (yet heuristic) approach is to choose the one with a smaller objective function value. A previous study (Nguyen, Ong, and Krasnogor, 2007) has already shown that the LS improvement over best individuals can result in good algorithmic performance. Note that the EA is supposed to be developed with good exploration ability. Therefore, we may postulate that the current best solution is more possible to be located in the attraction basin of a high-quality optimum than the other visited solutions. The stagnation of current evolutionary search also implies that the EA has reached the fine search stage. The use of the LS will make the exploitation much more efficient than the EA itself. Since we do not need the EA to carry out exploitation, T should be relatively small. A small T can also avoid the EA expending computational resources in cases where a large attraction basin is encountered.

In the algorithmic framework, at the end of each cycle, the current best solution (or cycle best, i.e., $\mathbf{x}_c$) is used as the initial solution for the LS. During the search, we may think of not applying the LS to improve $\mathbf{x}_c$ in cases where its fitness is worse than previous ones (i.e., $\mathbf{x}_j, 0 \leq j \leq c-1$). However, it is still worth expending computational cost on improving $\mathbf{x}_c$ since we do not have any criteria to tell its location in the fitness landscape. Alternately, some criteria to decide intelligently whether to apply the local search could be developed. For example, if the minimal distance between $\mathbf{x}_c$ and $\mathbf{x}_j, 0 \leq j \leq c-1$ is less than a threshold, then it may not be wise to apply local search to improve $\mathbf{x}_c$ since it may lead to the same local optimum. We do not include this consideration in our implementation, since there are no theoretical rules to decide the threshold: the threshold will depend on the optimisation problem.

If the stop criterion has not been met, we either maintain the current search, or restart a new search. To restart the search in the $(c+1)$th cycle, a new population of solutions can be randomly or heuristically generated. Note that as we already have some knowledge from previous search history, this may be useful for further searches in two ways. First, the new population can be generated intelligently by taking this knowledge into consideration. For example, we may want the generated population to be far away from the local optima identified so far, so that further searches can escape from them. Second, this knowledge may be used to guide the creation of offspring in the future searches. We will address these issues in the following sections.

To summarise, we see that developing an efficient MA based on the developed framework mostly depends on three factors, that is, an EA with effective exploration capability, a proper LS, and an intelligent diversification scheme. Among these factors, the diversification scheme is expected to help the search escape from local optima through the use of the learned knowledge if the EA fails to provide a solution that locates in the attraction basin of the global optimum. This implies that the developed MA may not perform well on optimisation problems if the learned knowledge has no prospective guidance on the global optimum, for example, problems with very unstructured fitness landscapes.

Algorithm 2 The EDA framework

1. **Initialization**. Randomly generate a set of solutions as the initial population $\mathcal{P}(0)$; Set $t := 0$;
2. **repeat**
3. **Selection**. Select promising solutions from $\mathcal{P}(t)$ to constitute the parent set $\mathcal{P}^s(t)$;
4. **Modeling**. Build a probability model $p(\mathbf{x}, t)$ from $\mathcal{P}^s(t)$;
5. **Sampling**. Sample offspring from $p(\mathbf{x}, t)$ to constitute a set S;
6. **Replacement**. Replace all or partially $\mathcal{P}(t)$ with S to form $\mathcal{P}(t+1)$; set $t := t+1$;
7. **until** stop criterion has been met.

Algorithm 3 The adaptive univariate probability model

Input: The parent set $\mathcal{P}^s(t)$.
Output: The probability $p(x_i, t)$ for the i-th component x_i.
1. Find $\ell_i^{\min} = \min\{x_i^k(t), 1 \le k \le K\}$, and $\ell_i^{\max} = \max\{x_i^k(t), 1 \le k \le K\}$;
2. Assign a small probability value to the intervals $[\ell_i^{\min} - \varepsilon_i, \ell_i^{\min}]$ and $[\ell_i^{\max}, \ell_i^{\max} + \varepsilon_i]$, and a big probability value to $[\ell_i^{\min}, \ell_i^{\max}]$, where ε_i is a predefined small positive real number.

3 Exemplar Algorithmic Implementation

The previous section describes a generic algorithmic framework. In this section, we describe one specific implementation of the framework in which our implementation is based on an estimation of distribution algorithm (EDA). In this section, we describe the various components that form our implementation, including an adaptive multivariate model and an offspring generation scheme, and the embedding of these components. To begin, we briefly describe the EDA.

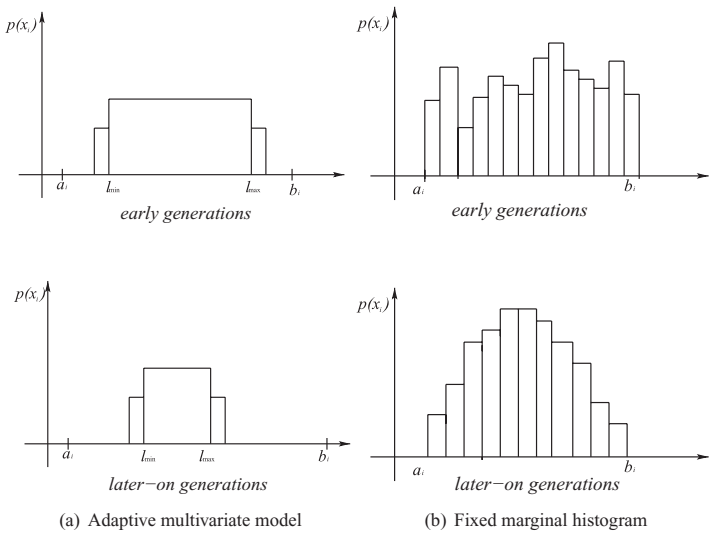
3.1 Estimation of Distribution Algorithm

The EDA was initially proposed by Mühlenbein and Paaß (1996). It maintains and evolves a set of individuals as in other EAs. But EDAs do not use crossover or mutation operators to produce new offspring. Instead, EDAs produce offspring by sampling from a probabilistic model. The statistical information extracted from promising solutions is used to construct the probabilistic model. The sampling of new offspring from the probability model is expected to guide the search to promising search areas. EDAs can be summarised as shown in Algorithm 2.

Existing EDAs for continuous optimisation can be classified with respect to the statistical model $p(\mathbf{x}, t)$ assumed. The probability models assumed include a Gaussian distribution (Larrañaga and Lozano, 2002), a Gaussian mixture (Bosman and Thierens, 2000), and a histogram (Tsutsui et al., 2001; Zhang et al., 2003). Readers are referred to Larrañaga and Lozano (2002) for a detailed description of EDAs.

3.2 Adaptive Multivariate Model

In our exemplar algorithm, an adaptive multivariate probability model $p(\mathbf{x}, t) = \prod_{i=1}^{n} p(x_i, t)$ with $\mathbf{x} = (x_1, \ldots, x_n)$ is assumed to represent the statistical information extracted from the parent set. Suppose that at generation t, the parent set consists of $\mathcal{P}^s(t) = \{\mathbf{x}^1(t), \mathbf{x}^2(t), \ldots, \mathbf{x}^K(t)\}$ where K is the size of the parent set. For the ith dimension, $p(x_i, t)$ is computed as described in Algorithm 3.



(a) Adaptive multivariate model (b) Fixed marginal histogram

Figure 2: (a) The adaptive multivariate probability model and (b) the univariate marginal histogram model at early and later stages.

The distributions of the ith variable, x_i, in the early and later stages of the search could be the profile as shown in Figure 2(a). In comparison, the marginal histogram model developed in Zhang et al. (2003) could be the profile as shown in Figure 2(b). In the marginal histogram model, the range of each variable is divided into a fixed number of subintervals with the same length. The histogram of the selected solutions at each variable is used to represent the probability model. The difference between the developed model and the histogram model is that the range of the decision variable in the developed model will adapt to the search environment.

From Figure 2, we see that the flat distribution in the adaptive multivariate probability model could promote exploration of the search space. On the other hand, the developed model will inevitably shrink the search space, and so accelerate the speed of evolutionary search. However, the shrinkage will weaken the exploration capability, since there is no chance to explore the search space outside $[\ell_i^{\min}, \ell_i^{\max}]$ whenever the shrinkage has happened. To moderate this problem, we adopt two strategies. First, we expand the present search space to $[\ell_i^{\min} - \varepsilon_i, \ell_i^{\max} + \varepsilon_i]$, where ε_i is a small number which is proportional to $b_i - a_i$. Second, the uniform distribution is applied to the interval $[\ell_i^{\min}, \ell_i^{\max}]$ (step 2). This is to make the generated solutions as diversified as possible in the search space, so that a good exploration can be expected. The time complexity to compute $p(\mathbf{x}, t)$ is $\mathcal{O}(nK)$, which is linear to the dimension and the selected population size.

3.3 Guided Mutation and Multiple Sampling Strategy

The proximate optimality principle (POP), first stated by Glover and Laguna (1998), has been applied widely (explicitly or implicitly) in meta-heuristics proposed for combinatorial optimisation problems, such as in Zhang et al. (2005) for the maximum clique problem, in Stützle (1999) and Zhang et al. (2004) for the quadratic assignment problem, and many others. It has been found in Glover (1998) that optimal solutions of the traveling salesman problem have 80% similarities. The POP states that good solutions

Algorithm 4 The guided mutation operator (note that $[A]$ rounds the elements of A to the nearest integers greater than or equal to A, and n is the dimension of the solution vector).

Algorithm $(\mathbf{x}^*, \alpha, p(\mathbf{x}))$
Input: A template solution $\mathbf{x}^* = (x_1^*, \cdots, x_n^*)$, a real number $0 \leq \alpha \leq 1$ and a probability model $p(\mathbf{x}, t) = \prod p(x_i, t)$.
Output: An offspring $\mathbf{x} = (x_1, \cdots, x_n)$.
1. Set $U = \{1, 2, \cdots, n\}$ and $d := \lceil \alpha n \rceil$; Randomly select a set of indices $V \subset U$ with $|V| = d$;
2. For all $i \in V$, set $x_i := x_i^*$; For all $j \in U \setminus V$, sample a value y from the probability model $p(x_j, t)$, set $x_j := y$;
3. Return $\mathbf{x}$;

have similar structures (components). The POP has also been successfully applied to continuous optimisation (Sun et al., 2005), which indicates that the POP can be very helpful in real-parameter optimisation. Actually, the POP has been widely but implicitly applied in EAs, such as in the mutation operators.

The explicit forms of the application of the POP are guided mutation (Zhang et al., 2005, 2004) and iterated local search (Stützle, 1999). Guided mutation has the advantage of generating offspring based on the guidance of statistical information. It has been particularly and most successfully applied to combinatorial optimisation problems. However, it can be easily applied to continuous optimisation problems in cases in which a fully factorised multivariate probability model is used. The offspring generation scheme can be summarised in Algorithm 4 with a template solution $\mathbf{x}^*$ and a probability model $p(\mathbf{x}, t)$.

To create offspring with Algorithm *GMSampling*, the best solution $\mathbf{x}^*$, that is, the solution with the least objective function value among all the visited solutions, is taken as the template solution. The fixing of some components (subset V) of $\mathbf{x}^*$ takes advantage of the legacy of the best solution. The creation of the rest of the components (subset $U \setminus V$) is guided by the probability model $p(\mathbf{x}, t)$. The random selection of V makes it possible for the search to find components in $\mathbf{x}^*$ that are the same as the global or near-global optimum. The time complexity of sampling one individual is $\mathcal{O}(n)$.

Parameter α, taking values from 0 to 1, controls the intensity of the fixing of the decision variables. It reflects to what extent POP holds for a certain optimisation problem. It can be expected that a large α value implies a quick evolutionary search, but a weak exploration capability, since a large number of variables will be copied from the best solution found so far. Moreover, the optimal α setting for efficient exploration varies with optimisation problems, and there is no a priori guidance on how to choose the best value in advance. Therefore, we propose to learn the parameter adaptively during the search in our implementation.

Recalling the weakness of the adaptive multivariate model described in Section 3.2, here we moderate this weakness through a multiple sampling strategy. Note that in almost all probability model based evolutionary algorithms, the number of individuals sampled from the probability model is usually less than, or equal to, the population size. As known in statistics (e.g., Markov Chain Monte Carlo theory, Robert and Casella, 2004), to truly represent the distribution of $p(\mathbf{x}, t)$, a large set of data points needs to be sampled from $p(\mathbf{x}, t)$. Therefore, statistically speaking, the small sample size applied in most EDAs will result in high sampling noise, leading to the false guidance problem in

the search procedure. That is, the sampling noise may mislead the search to possibly wrong areas, even though $p(\mathbf{x}, t)$ is accurate. Once this has happened, it is then hard to change back to the right track and, importantly, computational effort will be wasted. One straightforward way to reduce the sampling noise is to sample many more individuals to carry out replacement for the construction of the follow-up generation. This is the so-called multiple sampling strategy.

It should be pointed out that if optimisation problems do not obey the POP assumption (e.g., optimisation problems with discontinuous search domains), that is, if the commonality among the local optima and the global optimum does not exist, the later search cannot obtain any benefit from the best solution found so far. Consequently, the search may be inefficient, since it will rely on EA and LS entirely.

3.4 The Specific Algorithm

In this section, we embed the developed components described in Sections 3.2–3.3 to build the algorithm used in this paper.

3.4.1 The EDA

To construct an EDA, as seen in Algorithm 2, we first need to choose a selection method. Here, we apply the well-known truncation selection. In truncation selection, the fitness values of individuals in the population are sorted, and the individuals with the best fitness values are selected. We use the adaptive multivariate probability model described in Algorithm 3 to carry out modeling. The sampling method described in Algorithm 4 is applied to create new offspring. Note that there is a control parameter $\alpha \in [0, 1]$ in the sampling method. We fix it in each cycle and adjust it adaptively at the end of each cycle as follows:

if the current cycle found a better solution $\mathbf{x}_c^*$ than the previous ones $\mathbf{x}_j^*$, $1 \leq j \leq c-1$

then set $\alpha := \alpha + \frac{1}{n}$;

else set $\alpha := \max\{\alpha - \frac{1}{n}, 0\}$;

end if

That is, when the current cycle has found a better solution ($f(\mathbf{x}_c^*) < f(\mathbf{x}_j^*), 0 \leq j \leq c-1$), we will set a bigger α value for the following cycle. The reasoning behind this is that we have more confidence that there are more similar components between the best solution $\mathbf{x}^* = \arg\min_{0 \leq i \leq c}\{f(\mathbf{x}_1^*), \ldots, f(\mathbf{x}_c^*)\}$ and the global optimum under the POP assumption. If the current cycle cannot find a better solution, then it means our choice of α is not proper. In this case, α decreases by $\frac{1}{n}$ (ensuring α is greater than zero). If no better solution is found in the new cycle of search, α will be decreased again. Obviously, α cannot be negative. In the first cycle, α is set to zero. This is to make the algorithm explore the search space as much as possible. Moreover, if in some consecutive cycles, there are no better solutions $\mathbf{x}_c^*$ found, α could decrease to zero, which means that new search areas are to be explored.

As discussed in Section 3.3, it is appropriate to sample a large number of offspring to reduce the sampling noise. In our implementation, we first introduce an indicator vector $I = \{i_1, \ldots, i_M\}$ in Selection. That is, if a solution j is selected, $i_j := 1$, otherwise $i_j := 0$. We combine the sampling and replacement together as show in Algorithm 5.

Algorithm 5 Sampling and replacement

for $j = 1$ **to** M
  if $i_j = 0$, **then**
    Sample L solutions from $p(\mathbf{x}, t)$ using Algorithm *GMSampling*($\mathbf{x}^*$, α, $p(\mathbf{x}, t)$);
    Evaluate the sampled individuals, and pick the best one to replace $\mathbf{x}_j$;
  else
    Keep the current solution $\mathbf{x}_j$.
  end if
end for

3.4.2 The Restart Mechanism

In our implementation, we apply a simple diversification scheme. That is, we randomly generate a set of individuals with size $N = \max\{1000, 2n\} \gg M$. The M individuals that have the largest distances to $\mathcal{X} = \{\mathbf{x}_0^*, \ldots, \mathbf{x}_c^*\}$ are selected to form the initial population of the $(c+1)$th cycle. The distance between a randomly generated individual $\mathbf{x}$ and the set of visited local optima $\mathcal{X}$ is defined as:

$$d_{\mathbf{x}} = \min_{0 \le i \le c} \|\mathbf{x} - \mathbf{x}_i^*\|$$

where $\|\cdot\|$ is the Euclidean distance (but other distance measures may be used).

3.4.3 The Local Search

To carry out local search, we use an iterative derivative-free unconstrained optimisation algorithm developed by Powell (2006, 2008), called NEWUOA. Of course, we do not rule out the application of other local search algorithms.

Essentially, NEWUOA is an iterative trust-region method. At the kth step, it constructs a local quadratic model $\mathcal{Q}_k$ by interpolating a set of m solutions (initially sampled in the radius ρ_k ($\rho_0 = \rho_{\text{beg}}$) of the initial solution $\mathbf{x}_{\text{ini}}$). The local model within the trust region, which is a disc of radius $\Delta_k = \rho_k$, is optimized. Based on the quality of the obtained optimal solution $\mathbf{x}^*$, the trust region radius Δ_k is either expanded or contracted. The local model is updated by minimizing the Frobenius norm of the change to the second derivative of $\mathcal{Q}_k$. The local improvement continues until the step size $\delta = \|\mathbf{x}_k - \mathbf{x}^*\| < \rho_k$, in which case we decrease ρ_k. The algorithm terminates if ρ_{end} is reached.

From this brief description of NEWUOA, we can see that the parameters of NEWUOA are ρ_{beg}, ρ_{end}, and an integer $m \in [n+2, \frac{1}{2}(n+1)(n+2)]$, apart from the initial solution $\mathbf{x}_{\text{ini}}$. $m = 2n+1$ is used in our application (as recommended in Powell, 2008), ρ_{beg} is set as $0.2 \max\{|\mathbf{x}_{\text{ini}}|\}$, and ρ_{end} (which can be used to control the final accuracy of the local optima) is set as 10^{-10}. NEWUOA does not need any gradient information and hence is applicable to a large set of continuous optimisation problems. Readers are referred to Powell (2008) for a detailed description of NEWUOA.

4 Experimental Studies

In this section, we experimentally study the performance of IMMA. The experimental studies are divided into three parts. First, we carry out experiments to study the effects of the algorithmic components and parameters on the performance of IMMA. The algorithmic parameters include the population size M, the selected population size K, the sampling multiplication number L, the restart criterion number T, and the

expansion parameter ε_i. The expansion in each dimension of the search space is set to be $\frac{b_i - a_i}{M}$, $1 \leq i \leq n$. This is set to eliminate the influence of the variable scales in different coordinates. Among these parameters, we found that the most important ones are M and L, while K and T can simply be set as $K = M/2$ and $T = 5$, respectively.

Second, we compared the developed algorithm with a diverse set of evolutionary algorithms in literature. Finally, scalability analysis was carried out to study the effect of the dimensionality of the problem on the performance of the algorithm.

4.1 Test Suite and Performance Evaluation Criteria

We test our algorithm on a set of 40 commonly used test functions in the literature. Appendix A lists the descriptions of the functions and their respective characteristics including the global optimum f^*, dimensionality, and the characteristics, such as epistasis, modality, and discontinuity. The test functions f_1 through f_{10}, f_{26} through f_{34}, and f_{37}, which are the same as used in Noman and Iba (2008), are applied to carry out the component analysis. In component analysis, to evaluate the performance of the algorithms, the performance criteria applied in Noman and Iba (2008) and Suganthan et al. (2005) are adopted. The analysis is based on the statistics of the criteria within 25 trials. In all cases, the maximal number of fitness evaluations (NFE) to carry out the trials is $10{,}000 \times n$ where n is the dimension of the problem. The criteria can be summarised as follows:

- **Error Criterion**. The function error value (FEV) is defined as $f(\mathbf{x}) - f(\mathbf{x}^*)$, the difference between the objective values (fitness) of the global optimum of the function and a solution $\mathbf{x}$. We record the minimal FEV that the algorithm can find within the maximal NFE, and calculate the mean (AVG_E) and standard deviation (SD) of these FEVs. The notation $\text{AVG}_E \pm SD_E$ is used to summarise the criterion.

- **NFE Criterion**. If the minimal FEV is less than an accuracy level ϵ within NFE, the trial is considered to be successful. We count the number of successful trials, and denote the number as SUC. The number of fitness evaluations required for the algorithm to be successful is recorded, and the mean and standard deviation are calculated, and denoted by AVG_N and SD_N. The accuracy level ϵ is fixed to 10^{-6} for f_1 through f_5 and f_{26} through f_{40}, to 10^{-2} for f_6 through f_{16} and to 10^{-1} for f_{17} through f_{25} as used in Suganthan et al. (2005). The notation used here is $\text{AVG}_N \pm SD_N$.

- **Convergence Graphs**. The graphs show the evolution process of the algorithms, that is, the average error performance against the generations and/or NFEs.

- **Hypothesis Test.** Hypothesis test (either the rank sum test or the Z test) is applied in the following experiments at the 5% significance level. A p value less than .05 suggests a significant difference.

NFEs used by IMMA include the NFEs consumed by NEWUOA and the developed EDA. The performance comparisons of the various EAs used varying performance criteria which will be described specifically in each context.

4.2 Component Analysis

4.2.1 Sensitivity to Population Size

The population size is an important factor to the performance of EAs. Large population sizes tend to encourage the preservation of diversity and hence improve the exploration

ability of EAs. However, a large population size will make EAs exhaust the fitness evaluations too quickly. Therefore, there should be a trade-off between the exploration capability and the computational cost used for the exploration. To investigate the effect of the population size on the IMMA performance, we apply the IMMA with varied population size on the test functions with $n = 30$. The adaptive multivariate model and the multiple sampling strategy with $L = 3$ are applied. The population size is set as $M = kn$, where $k \in \{1, 2, 5, 10\}$. The experimental results are listed in Table 1, where best results are shown in bold type. The upper part of the table lists the NFE criterion obtained for those functions in which all runs reach the accuracy levels, while the lower part of the table shows the error criterion obtained for those functions that IMMA cannot reach the accuracy level in all 25 trials. Table 2 lists the average number of cycles in which IMMA reaches the accuracy level, the number of success trials, and the p values obtained from the rank sum test between the results for $M = 60$ and other population sizes.

The upper part of Table 1 indicates that the larger the population size, the more NFEs are required to reach the accuracy level. Moreover, the lower part of the table shows that (1) an IMMA with a small population size ($M = 30, 60$) can find better solutions than an IMMA with a large population size except for $f_{8,32}$ where the IMMAs with different M obtain similar results; (2) for $f_{30,34}$, the IMMA with a small population size achieves a higher success rate than the IMMA with a large population size ($M = 150, 300$). These observations show that a small population size is preferred over a large population size.

Moreover, comparing the IMMAs with population sizes $M = 30$ and 60 (i.e., $k = 1, 2$), we see that the IMMA with $k = 1$ requires less computational cost to reach the accuracy level except for $f_{10,29,33}$. However, the computational cost of the IMMA with $k = 2$ is still very competitive. Table 1 also indicates that the IMMA with $k = 2$ obtains better optimal solutions than the IMMA with $k = 1$ on the functions that cannot be solved successfully. Note that most of the functions in the upper part of Table 1 have no epistasis except for $f_{6,7,10,37}$, while most of these functions in the lower part have high epistasis (see Table A1 in the appendix). This shows that the IMMA with $k = 2$ can deal with high-epistasis functions better than the IMMA with $k = 1$. Hence, $k = 2$ should be preferred in case we do not have any prior knowledge about the characteristics of the optimisation problems.

It can be seen from the upper part of Table 2 that the number of cycles has a tendency to decrease along with an increase of population size. This indicates that, with a large population, a large number of iterations and thereafter a large number of fitness evaluations are required to complete a cycle. Consequently, only a small number of cycles (equivalently, local searches) can be carried out within a fixed number of fitness evaluations. The superior performance of the IMMA with a small population size implies that more local searches are required in order to achieve better performance.

Figure 3 displays the convergence graphs of IMMA on some selected functions. On the x axis is the number of fitness evaluations, and on the y axis is the FEVs. From Figure 3, we can see that usually small population sizes ($M = 30, 60$) require less NFEs to reach the accuracy level (e.g., Figure 3[a,b,c]) than with large population sizes ($M = 150, 300$), while Figure 3(d) shows that small population sizes usually obtain better FEV.

4.2.2 Effects of Selection

Another important factor affecting the performance of EAs is the selection operator. Note that under the IMMA framework, we need to speed up the exploration of EA. The selection method plays an important role in the convergence performance of EA. In

Table 1: The experimental results obtained by the IMMA with varying population size M for the test functions with $n = 30$, while NFEs are shown for the functions that can be solved successfully in the 25 trials, and the best error values are shown for the hard functions.

Function	$M = 30$	$M = 60$	$M = 150$	$M = 300$
$\text{AVG}_N \pm SD_N$				
f_1	**2.86e+03** ± 2.08e+02	6.55e+03 ± 6.27e+02	1.52e+04 ± 5.83e+03	3.03e+04 ± 1.17e+03
f_2	**8.68e+03** ± 9.55e+02	1.01e+04 ± 1.03e+03	1.33e+04 ± 1.75e+03	1.90e+04 ± 3.70e+03
f_3	**2.95e+04** ± 2.05e+03	3.13e+04 ± 2.57e+03	3.32e+04 ± 2.24e+03	3.80e+04 ± 2.83e+03
f_6	**2.10e+04** ± 1.84e+04	3.11e+04 ± 3.03e+04	4.13e+04 ± 3.02e+04	6.01e+04 ± 2.97e+04
f_7	**6.55e+03** ± 4.09e+03	1.02e+04 ± 6.28e+03	5.11e+04 ± 1.06e+04	9.61e+04 ± 5.16e+04
f_9	**3.79e+04** ± 1.12e+04	3.94e+04 ± 1.46e+04	3.90e+04 ± 9.43e+03	4.41e+04 ± 7.93e+03
f_{10}	5.47e+04 ± 1.45e+04	4.20e+04 ± 1.61e+04	**3.97e+04** ± 1.63e+04	5.19e+04 ± 1.64e+04
f_{28}	**2.87e+03** ± 1.36e+02	6.30e+03 ± 3.56e+02	1.85e+04 ± 1.36e+03	4.05e+04 ± 3.70e+03
f_{29}	2.83e+04 ± 8.77e+03	**2.67e+04** ± 6.21e+03	3.93e+04 ± 9.28e+03	5.09e+04 ± 1.72e+04
f_{31}	**4.35e+04** ± 1.19e+04	4.69e+04 ± 1.46e+04	5.60e+04 ± 1.66e+04	6.81e+04 ± 1.23e+04
f_{33}	2.73e+04 ± 1.23e+04	**2.02e+04** ± 9.15e+03	2.67e+04 ± 8.68e+03	4.28e+04 ± 3.43e+03
f_{37}	**1.19e+04** ± 7.70e+03	1.93e+04 ± 1.65e+04	4.56e+04 ± 5.93e+03	6.85e+04 ± 4.63e+04
$\text{AVG}_E \pm SD_E$				
f_4	1.31e+05 ± 2.35e+04	**2.94e+04** ± 6.16e+03	5.39e+04 ± 3.59e+04	6.23e+04 ± 5.35e+03
f_5	1.67e+03 ± 3.78e+02	**1.68e+03** ± 3.27e+02	2.05e+03 ± 2.20e+02	2.20e+03 ± 1.97e+02
f_8	2.00e+01 ± 1.28e–10	2.00e+01 ± 1.12e–10	2.00e+01 ± 1.77e–10	2.00e+01 ± 1.37e–10
f_{26}	2.92e–01 ± 4.00e–02	**2.84e–01** ± 4.73e–02	3.80e–01 ± 7.07e–02	3.84e–01 ± 6.88e–02
f_{27}	**4.13e+01** ± 5.32e+01	**4.13e+01** ± 4.97e+01	1.31e+02 ± 7.13e+01	1.93e+02 ± 1.08e+02
f_{30}	**9.72e–14** ± 7.77e–14	1.63e–13 ± 3.26e–13	2.86e–03 ± 4.37e–03	4.24e–03 ± 5.88e–03
f_{32}	3.82e–04 ± 1.01e–12	3.82e–04 ± 7.00e–13	3.82e–04 ± 7.59e–13	3.82e–04 ± 1.34e–12
f_{34}	6.61e–13 ± 4.42e–13	**5.56e–13**± 4.21e–13	9.98e–08 ± 2.76e–08	8.79e–04 ± 3.04e–04

Table 2: The number of cycles and success trials to reach the accuracy level and the p values obtained from the rank sum test between the results obtained by the IMMA with $M = 60$ and the other population sizes.

					p_m		
Function	$M = 30$	$M = 60$	$M = 150$	$M = 300$	$M = 30{,}60$	$M = 60{,}150$	$M = 60{,}300$
Number of cycles							
f_1	1.0	1.0	1.0	1.0	**<.001**	**<.001**	**<.001**
f_2	1.0	1.0	1.0	1.0	**<.001**	**<.001**	**<.001**
f_3	1.0	1.0	1.0	1.0	**.011**	**.010**	**<.001**
f_6	1.3	1.0	1.0	1.0	0.167	**.045**	**.002**
f_7	2.1	1.5	1.0	1.0	**.008**	**<.001**	**<.001**
f_9	15.9	13.2	8.1	5.8	0.687	0.909	0.170
f_{10}	16.5	11.5	9.3	6.5	**.007**	0.620	**.042**
f_{28}	1.0	1.0	1.0	1.0	**<.001**	**<.001**	**<.001**
f_{29}	4.4	3.5	2.9	2.8	0.464	**.000**	**<.001**
f_{31}	14.1	10.1	7.9	6.6	0.376	.051	**<.001**
f_{32}	20.3	15.8	12.1	10.5	**<.001**	**.001**	**<.001**
f_{33}	3.8	2.6	2.5	1.1	**.029**	**.017**	**<.001**
f_{37}	3.5	2.1	1.6	1.2	.053	**<.001**	**<.001**
Number of success trials							
f_4	0	0	0	0	**<.001**	**.003**	**<.001**
f_5	0	0	0	0	0.843	**<.001**	**<.001**
f_8	0	0	0	0	1.000	1.000	1.000
f_{26}	0	0	0	0	0.525	**<.001**	**.000**
f_{27}	0	0	0	0	1.000	**<.001**	**<.001**
f_{30}	25	25	17	15	0.336	**.003**	**.001**
f_{34}	25	25	25	23	0.398	**<.001**	**<.001**

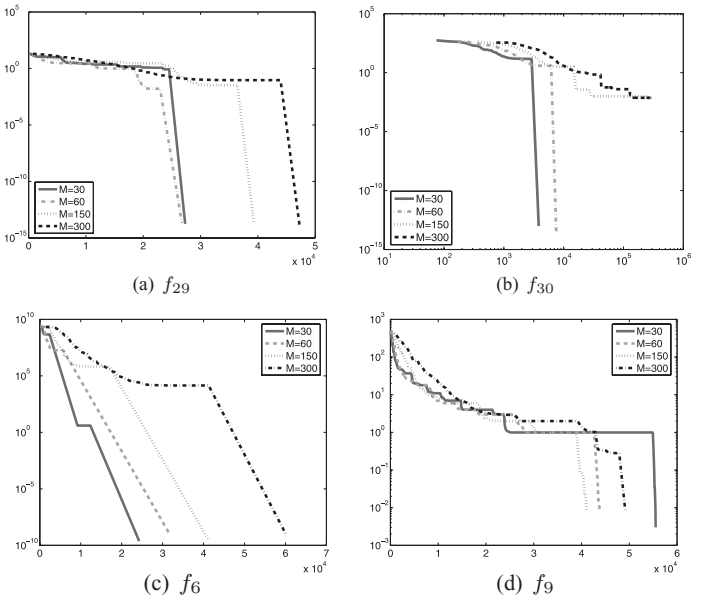


Figure 3: Convergence curve of IMMA with variations of population size for selected functions ($n = 30$). The x axis shows the number of fitness evaluations. The y axis shows the error values.

this section, we experimentally compare two algorithms, IMMA-P (with proportional selection) and IMMA-T (with truncation selection) on the functions with $n = 30$ to carry out analysis of different selection operators. In our implementation for fitness proportional selection, the probability that an individual with fitness f is selected is proportional to $\exp(-f)$ (in case of minimization).

To apply the two algorithms, the parameters are set as $M = 60$ and $L = 3$. Table 3 lists the best error criterion and the NFE criterion, respectively, where best results are shown in bold type. In the table, columns p_v and p_n list the p values of the rank sum test on the best error criterion and the NFE criterion, respectively. From the table, it can be seen that at a 5% significance level, IMMA-P reaches better performances than IMMA-T on only one test function, f_{37}, while on three functions, $f_{2,8,29}$, the two algorithms perform similarly. Thus, we can conclude that IMMA-T outperforms IMMA-P on average in terms of solution quality and computational cost. The experimental result justifies that truncation selection is significantly better than proportional selection under the IMMA framework. Figure 4 shows the convergence curve comparing IMMA-P and IMMA-T on some selected functions. The plots show that IMMA-T has improved convergence characteristics over IMMA-P.

4.2.3 Study on the Multiple Sampling Strategy

In this section, we experimentally analyze the effect of the multiple sampling strategy on the performance of IMMA. To carry out this study, we apply the IMMA with different sampling number L on the functions. We are interested in whether the multiple sampling strategy is beneficial to the exploration capability of IMMA. The algorithmic parameter settings in the experiments are $M = 60$. L is varied from 1 to 5. Table 4 lists the experimental results obtained by the IMMA with different L settings in terms of the best error values and NFEs. In the table, the best results are shown in bold type. The p values of the rank sum test on the results of $L = 2$ and $L = 3$ against the other L values are listed in Table 5. From Table 5 we see that:

1. On the functions that cannot be solved successfully, the IMMA with sampling size $L = 1$ has the best performance in terms of error value on only one function (f_{27}), but this performance difference is not significant. Moreover, it can be seen that the IMMA with a large sampling size ($L = 4, 5$) does not necessarily result in better FEVs.
2. In terms of efficiency (NFEs), the IMMAs with $L = 2, 3$ require fewer fitness evaluations to reach the accuracy level on all the multimodal functions except $f_{6,31}$ (functions f_1 through f_3 and f_{28} are unimodal).

From these observations, we see that relatively more individuals (e.g., $L = 2, 3$) sampled from the probability model can indeed improve exploration capability on average. However, a large L (e.g., 5) usually needs more fitness evaluations to reach the accuracy level. Therefore, we may conclude that the multiple sampling strategy with a moderate L can favour evolutionary search. Since it is difficult to develop a mathematical model to determine an optimal L for the balance between the exploration capability and the computational cost, we have to rely on experimental study. As seen from the results, the IMMA with $L = 2, 3$ achieves better results on almost all the functions. Hence, in practice, we can simply start our search for optimal L from 2.

Table 3: Evaluation criteria for fitness proportional and truncation selection at $n = 30$. IMMA-P denotes the IMMA with proportional selection, while IMMA-T denotes IMMA with truncation selection. Columns p_v and p_n list the p values of the rank sum test on the results of the best error criterion and the NFE criterion, respectively. — denotes that the rank sum test is not necessary either because both algorithms are fully successful (no p_v is available) or fully failed (no p_n is available). * denotes that the algorithm stops at the maximum NFEs, that is, 3.0×10^5.

	IMMA-P		IMMA-T			
Function	$AVG_E \pm SD_E$	$AVG_N \pm SD_N$(SUC)	$AVG_E \pm SD_E$	$AVG_N \pm SD_N$(SUC)	p_v	p_n
f_1	0.00e+00 ± 0.00e+00	7.20e+03 ± 4.74e+02(25)	.00e+00 ± 0.00e+00	**6.55e+03** ± 6.27e+02(25)	—	**<.001**
f_2	7.03e–12 ± 1.36e–11	**9.51e+03** ± 1.19e+03(25)	6.22e–12 ± 2.57e–12	1.00e+04 ± 1.03e+03(25)	—	.133
f_3	3.03e–08 ± 5.36e–07	3.95e+04 ± 2.12e+03(25)	1.51e–08 ± 1.54e–08	**3.13e+04** ± 2.57e+03(25)	—	**<.001**
f_{28}	2.11e–32 ± 4.00e–32	8.72e+03 ± 7.52e+03(25)	1.93e–34 ± 2.85e–34	**6.30e+03** ± 3.56e+02(25)	—	**<.001**
f_{29}	2.20e–14 ± 7.11e–15	2.78e+04 ± 1.66e+04(25)	2.11e–14 ± 7.80e–15	**2.67e+04** ± 6.21e+03(25)	—	.759
f_{30}	9.78e–14 ± 8.75e–14	9.14e+03 ± 7.23e+02(25)	1.63e–13 ± 3.26e–13	**7.68e+03** ± 3.28e+03(25)	—	**.040**
f_{32}	4.29e–04 ± 1.63e–04	3.13e+04 ± 1.47e+04(25)	3.82e–04 ± 7.00e–13	**1.47e+04** ± 3.74e+03(25)	—	**<.001**
f_{33}	8.42e–14 ± 2.26e–13	9.42e+04 ± 7.78e+04(25)	3.09e–14 ± 3.02e–14	**2.02e+04** ± 9.15e+03(25)	—	**<.001**
f_{37}	2.96e–10 ± 1.30e–09	**1.37e+04** ± 7.36e+03(25)	4.73e–11 ± 4.06e–11	1.93e+04 ± 1.65e+03(25)	—	**.001**
f_4	6.52e+04 ± 9.10e+03	*	**2.94e+04** ± 6.16e+03	*	**<.001**	—
f_5	2.61e+03 ± 8.78e+02	*	**1.68e+03** ± 3.27e+02	*	**<.001**	—
f_8	2.00e+01 ± 1.88e–10	*	2.00e+01 ± 1.12e–10	*	1.000	—
f_{26}	1.87e+00 ± 2.12e–01	*	**2.84e–01** ± 4.73e–02	*	**<.001**	—
f_{27}	5.00e+02 ± 9.51e+01	*	**4.13e+01** ± 4.97e+01	*	**<.001**	—
f_6	1.91e+03 ± 8.78e+02	*	**3.18e–10** ± 2.56e–10	3.11e+04 ± 3.03e+04(25)	**<.001**	**<0.000**
f_7	3.54e+04 ± 8.78e+03	*	**1.31e–12** ± 4.79e–12	1.02e+04 ± 6.28e+03(25)	**<.001**	**<.001**
f_9	2.43e+00 ± 2.17e+00	2.28e+05 ± 6.55e+04(7)	**1.09e–03** ± 3.03e–03	3.94e+04 ± 1.46e+04(25)	**<.001**	**<.001**
f_{10}	3.43e+01 ± 6.87e–01	2.59e+05 ± 9.85e+04(5)	**1.90e–03** ± 3.48e–03	4.20e+04 ± 1.61e+04(25)	**<.001**	**<.001**
f_{31}	3.46e+00 ± 2.26e+00	2.90e+05 ± 3.79e+04(2)	3.69e–11 ± 1.10e–10	**4.69e+04** ± 1.46e+04(25)	**<.001**	**<.001**
f_{34}	2.41e–02 ± 5.22e–02	2.21e+05 ± 5.05e+04(13)	5.56e–13 ± 4.21e–13	**4.02e+04** ± 2.22e+04(25)	**.030**	**<.001**

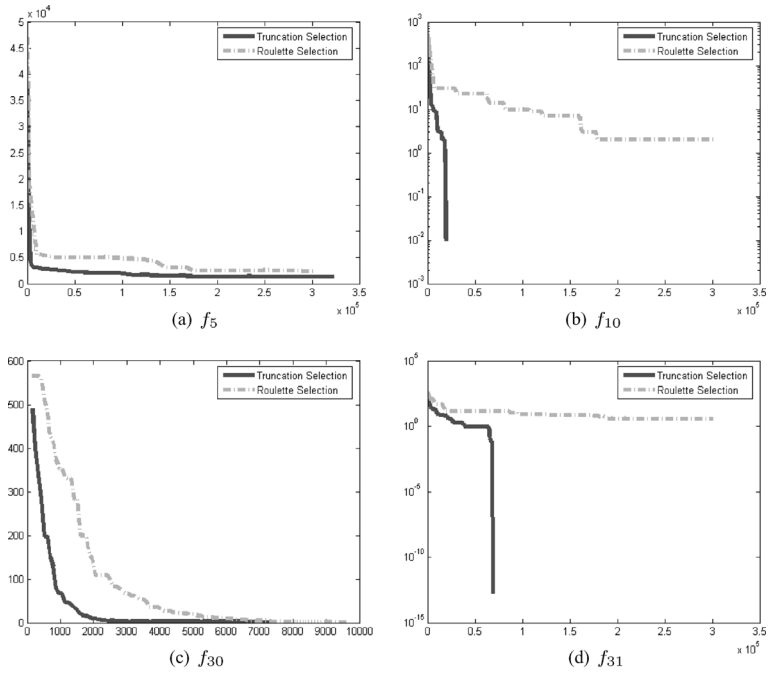


Figure 4: Convergence curves of the IMMA with different selection operations for selected functions $f_{5,10,30,31}$ ($n = 30$). The x axis shows the number of fitness evaluations. The y axis shows the best error values.

4.2.4 The Superiority of the Adaptive Model

In this section, we investigate the effect of the adaptive multivariate model on the performance of IMMA. We compare two algorithms, IMMA-H and IMMA, to carry out the investigation. Both algorithms have the same algorithmic framework as described in Section 3.4. The difference is that in IMMA-H, the histogram model (Zhang et al., 2003) is embedded. The other parameters are set as $M = 60$ and $L = 3$. The comparison results are listed in Table 6, in which the best results are shown in bold type. In cases for which both algorithms reached the accuracy level in 25 trials, the rank test is performed on the NFE results and summarised in the p_{nh} column. On the other hand, if both algorithms failed to reach the accuracy level in all 25 trials, the rank test is performed on the best error values and summarised in the p_{vh} column.

From Table 6, we see that at the 5% significance level, in terms of the best error values, IMMA performs significantly better than IMMA-H on 10 functions that cannot be solved successfully, except for f_8, where the two algorithms perform similarly. In the other 10 functions that can be successfully solved, IMMA-H performs significantly better than IMMA on five functions (f_6, f_{28} through f_{30}, and f_{37}) in terms of NFEs, and IMMA is significantly better than IMMA-H on the other five functions. This shows the superiority of the adaptive multivariate model against the histogram model, on average. The graphs in Figure 5 show that the IMMA with the adaptive model exhibits a higher convergence speed than that of IMMA-H, except for function f_{29}. This also confirms that the developed adaptive multivariate model is better than the histogram model in generating promising solutions.

Table 4: NFEs obtained for the selected functions with $n = 30$ that can be successfully solved and the best error values for the functions that cannot be solved by the IMMA with varied sampling size L.

Function	$L = 1$	$L = 2$	$L = 3$	$L = 4$	$L = 5$
$AVG_N \pm SD_N$					
f_1	**3.08e+03** ± 5.35e+02	5.15e+03 ± 3.28e+02	6.55e+03 ± 6.27e+02	8.00e+03 ± 5.43e+02	9.31e+03 ± 5.46e+02
f_2	1.28e+03 ± 8.01e+02	1.38e+03 ± 1.17e+03	**1.01e+04** ± 1.03e+03	1.16e+04 ± 2.38e+03	1.26e+04 ± 3.71e+03
f_3	3.21e+04 ± 2.63e+03	**3.04e+04** ± 2.11e+03	3.13e+04 ± 2.57e+03	3.27e+04 ± 3.48e+03	3.29e+04 ± 2.88e+03
f_6	**1.43e+04** ± 8.91e+03	2.18e+04 ± 1.60e+04	3.11e+04 ± 3.03e+04	4.74e+04 ± 4.94e+04	3.27e+04 ± 2.48e+04
f_7	1.71e+04 ± 5.88e+03	1.77e+04 ± 1.49e+04	**1.02e+04** ± 6.28e+03	6.19e+04 ± 7.96e+04	4.64e+04 ± 5.82e+04
f_9	4.11e+04 ± 1.38e+04	**3.51e+04** ± 1.07e+04	3.94e+04 ± 1.46e+04	3.99e+04 ± 1.12e+04	4.25e+04 ± 1.67e+04
f_{10}	4.96e+04 ± 1.61e+04	4.66e+04 ± 1.39e+04	**4.20e+04** ± 1.61e+04	4.63e+04 ± 1.57e+04	4.42e+04 ± 1.33e+04
f_{28}	**3.09e+03** ± 6.15e+02	4.83e+03 ± 3.64e+02	6.30e+03 ± 3.56e+02	7.93e+03 ± 4.23e+02	9.17e+03 ± 6.05e+02
f_{29}	2.69e+04 ± 6.39e+03	2.69e+04 ± 6.25e+03	2.67e+04 ± 6.21e+03	**2.64e+04** ± 5.91e+03	2.70e+04 ± 6.63e+03
f_{30}	2.65e+04 ± 9.18e+03	2.11e+04 ± 5.24e+04	**7.68e+03** ± 3.28e+03	3.57e+04 ± 6.89e+04	3.48e+04 ± 6.37e+04
f_{31}	**4.07e+04** ± 1.42e+04	4.40e+04 ± 9.10e+03	4.69e+04 ± 1.46e+04	5.24e+04 ± 1.83e+04	5.38e+04 ± 1.26e+04
f_{32}	1.48e+04 ± 3.64e+03	**1.37e+04** ± 2.56e+03	1.47e+04 ± 3.74e+03	1.72e+04 ± 3.56e+03	1.95e+04 ± 4.06e+03
f_{33}	3.02e+04 ± 1.60e+04	2.43e+04 ± 1.25e+04	2.02e+04 ± 9.15e+03	**2.01e+04** ± 9.74e+03	2.38e+04 ± 1.37e+04
f_{34}	3.91e+04 ± 2.00e+04	**3.79e+04** ± 2.67e+04	4.02e+04 ± 2.72e+04	3.79e+04 ± 1.49e+04	4.16e+04 ± 2.11e+04
f_{37}	2.04e+04 ± 4.34e+03	2.53e+04 ± 9.95e+03	**1.93e+04** ± 1.65e+04	2.12e+04 ± 1.41e+04	2.39e+04 ± 2.32e+04
$AVG_E \pm SD_E$					
f_4	2.10e+05 ± 1.94e+04	1.11e+05 ± 1.63e+04	**2.94e+04** ± 6.16e+03	2.80e+05 ± 3.18e+04	2.52e+05 ± 2.18e+04
f_5	2.47e+03 ± 5.46e+02	1.74e+03 ± 3.19e+02	**1.68e+03** ± 3.27e+02	1.77e+03 ± 3.17e+02	1.69e+03 ± 2.37e+02
f_8	2.00e+01 ± 1.18e–10	2.00e+01 ± 1.29e–10	2.00e+01 ± 1.12e–10	2.00e+01 ± 1.20e–10	2.00e+01 ± 1.86e–10
f_{26}	3.92e–01 ± 5.34e–02	3.40e–01 ± 5.00e–02	**2.84e–01** ± 4.73e–02	2.94e–01 ± 4.90e–02	2.96e–01 ± 4.36e–02
f_{27}	**3.75e+01** ± 4.11e+01	3.79e+01 ± 4.57e+01	4.13e+01 ± 4.97e+01	4.93e+01 ± 5.11e+01	3.92e+01 ± 4.72e+01

Table 5: The p values obtained through the rank sum test on the experimental results obtained by $L = 2$ and $L = 3$ with the other L values.

	$p_{L=2}$				$p_{L=3}$			
Function	$L=1$	$L=3$	$L=4$	$L=5$	$L=1$	$L=2$	$L=4$	$L=5$
f_1	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**0.000**	**<0.001**
f_2	0.727	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**0.008**	**0.003**
f_3	**0.019**	0.189	**0.009**	**0.002**	0.287	0.189	0.119	**0.049**
f_6	**0.049**	0.190	**0.022**	0.079	**0.014**	0.190	0.173	0.842
f_7	0.863	**0.028**	**0.012**	**0.025**	**<0.001**	**0.028**	**0.004**	**0.005**
f_9	0.098	0.242	0.134	0.074	0.682	0.242	0.899	0.497
f_{10}	0.493	0.291	0.932	0.539	0.110	0.291	0.356	0.605
f_{28}	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**
f_{29}	0.982	0.947	0.805	0.926	0.929	0.947	0.858	0.875
f_{30}	0.614	0.215	0.405	0.412	**<0.001**	0.215	0.053	**0.044**
f_{31}	0.330	0.418	0.052	**0.004**	0.142	0.418	0.251	0.084
f_{32}	0.257	0.292	**0.001**	**<0.001**	0.956	0.292	**0.025**	**<0.001**
f_{33}	0.160	0.195	0.200	0.898	**0.012**	0.195	0.987	0.279
f_{34}	0.858	0.769	0.997	0.587	0.877	0.769	0.721	0.831
f_{37}	**0.034**	0.135	0.253	0.782	0.752	0.135	0.662	0.433
f_4	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**
f_5	**<0.001**	0.518	0.742	0.535	**<0.001**	0.518	0.333	0.902
f_8	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
f_{26}	**0.002**	**<0.001**	**0.003**	**0.003**	**<0.001**	**<0.001**	0.470	0.360
f_{27}	0.974	0.803	0.414	0.922	0.771	0.803	0.580	0.880

4.3 The Superiority of the Decoupled Strategy

In this section, we aim to study the proposed decoupled hybridisation strategy against the intertwined hybridisation scheme. To carry out the study, we compared IMMA with intertwined MA, called WMMA. WMMA is basically the same as IMMA, except that NEWUOA is applied to the best solution found at every 10 generations. The modeling, reproduction, and multiple sampling strategy are the same as in IMMA.

Table 7 summarises the obtained results. The rank sum test was performed to compare the results. The p values (shown in columns p_{vw} and p_{nw}) are obtained for the comparison of the best error values and NFEs, respectively. In the table, — in the p_{vw} column denotes that the rank sum test is not performed because both algorithms reach the termination criterion and hence it is not necessary to carry out the test in terms of the best error values. In the p_{nw} column, if both algorithms cannot reach the termination criterion within the maximal NFEs, the test is also not necessary to carry out in terms of NFEs.

From the table, it can be observed that IMMA performs significantly better than the WMMA on eight functions for which both algorithms cannot reach the termination criterion, except for f_8, where the algorithms perform the same. Moreover, IMMA needs less NFEs than the WMMA on the functions that can be solved within maximal NFEs, except on five test functions ($f_{1,2,28,30,37}$). This shows that, on average, the proposed hybridisation strategy is superior to the intertwined hybridisation scheme.

4.4 Comparison with Known EAs

In this section, IMMA is compared with the following algorithms.

Table 6: Experimental results for IMMA-H and IMMA on the selected functions. The number in parentheses is the number of successful trials, while no parenthetical number means zero successful trials. The last two columns list the p value of the rank sum test on NFEs (p_{nh}) and the best error values (p_{vh}). — denotes that the rank sum test is not necessary because either both algorithms are fully successful (no p_{vh} is available) or fully failed (no p_{nh} is available), * denotes that the algorithm stops at the maximum NFEs, that is, 3.0×10^5.

Function	IMMA-H		IMMA		p_h	
	$\text{AVG}_E \pm SD_E$	$\text{AVG}_N \pm SD_N$(SUC)	$\text{AVG}_E \pm SD_E$	$\text{AVG}_N \pm SD_N$(SUC)	p_{vh}	p_{nh}
f_1	0.00e+00 ± 0.00e+00	6.89e+03 ± 3.33e+02(25)	0.00e+00 ± 0.00e+00	**6.55e+03** ± 6.27e+02(25)	—	**.028**
f_2	5.93e–12 ± 3.05e–12	**9.58e+03** ± 1.23e+03(25)	6.22e–12 ± 2.57e–12	1.01e+04 ± 1.03e+03(25)	—	.156
f_3	2.25e–08 ± 2.44e–08	**3.06e+04** ± 2.16e+03(25)	1.51e–08 ± 1.54e–08	3.13e+04 ± 2.57e+03(25)	—	.277
f_6	3.56e–10 ± 4.80e–10	**1.68e+04** ± 9.04e+03(25)	3.18e–10 ± 2.56e–10	3.11e+04 ± 3.03e+04(25)	—	**.033**
f_7	3.16e–03 ± 4.00e–03	7.08e+03 ± 4.53e+03(25)	**1.31e–12** ± 4.79e–12	1.02e+04 ± 6.28e+03(25)	—	.057
f_{28}	6.83e–33 ± 8.28e–33	**2.52e+03** ± 4.03e+02(25)	1.93e–34 ± 2.85e–34	6.30e+03 ± 3.56e+02(25)	—	**<.001**
f_{29}	3.60e–14 ± 5.65e–15	**1.17e+04** ± 6.39e+03(25)	2.61e–14 ± 7.80e–15	2.67e+04 ± 6.21e+03(25)	—	**<.001**
f_{30}	1.35e–13 ± 1.01e–13	**3.12e+03** ± 4.10e+02(25)	1.63e–13 ± 3.26e–13	7.68e+03 ± 3.28e+03(25)	—	**<.001**
f_{33}	9.82e–14 ± 2.14e–13	2.82e+04 ± 1.88e+03(25)	3.09e–14 ± 3.02e–14	**2.02e+04** ± 9.15e+03(25)	—	**<.001**
f_{37}	8.25e–11 ± 1.30e–10	**8.10e+03** ± 3.58e+03(25)	4.73e–11 ± 4.06e–11	1.93e+04 ± 1.65e+04(25)	—	**.003**
f_4	3.85e+05 ± 7.31e+04	*	**2.94e+04** ± 6.16e+03	*	**<.001**	—
f_5	5.37e+03 ± 1.08e+03	*	**1.68e+03** ± 3.27e+02	*	**<.001**	—
f_8	2.00e+01 ± 1.14e–10	*	2.00e+01 ± 1.12e–10	*	1.000	—
f_{26}	4.28e+00 ± 2.39e+00	*	**2.84e–01** ± 4.73e–02	*	**<.001**	—
f_{27}	5.32e+02 ± 4.98e+01	*	**4.13e+01** ± 4.97e+01	*	**<.001**	—
f_9	9.04e+01 ± 1.32e+01	*	**1.09e–03** ± 3.03e–03	3.94e+04 ± 1.46e+04(25)	**<.001**	**<.001**
f_{10}	2.36e+02 ± 2.23e+01	*	**1.90e–03** ± 3.48e–03	4.20e+04 ± 1.61e+04(25)	**<.001**	**<.001**
f_{31}	4.18e+01 ± 5.26e–01	*	**3.69e–11** ± 1.10e–10	4.69e+04 ± 1.46e+04(25)	**<.001**	**<.001**
f_{32}	2.66e–01 ± 2.26e–01	1.31e+05 ± 6.47e+04(5)	**3.82e–04** ± 7.00e–13	1.47e+04 ± 3.74e+03(25)	**<.001**	**<.001**
f_{34}	6.07e–02 ± 8.43e–02	1.09e+05 ± 1.04e+05(7)	**5.56e–13** ± 4.21e–13	4.02e+04 ± 2.72e+04 (25)	**.001**	**.004**

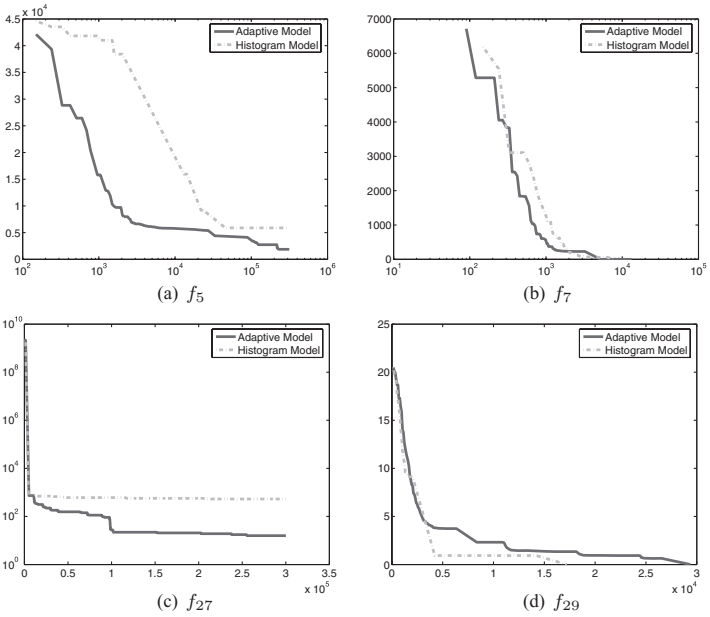


Figure 5: Convergence curve of the IMMA with different probability model for selected functions $f_{5,7,27,29}$ ($n = 30$). The x axis shows the number of fitness evaluations. The y axis shows the best error values.

1 **OGA/Q.** The orthogonal genetic algorithm with quantisation (OGA/Q; Leung and Wang, 2001), in which an orthogonal design is used to construct a crossover operator.

2 **HTGA.** The hybrid Taguchi genetic algorithm (HTGA; Tsai et al., 2004), in which a robust experimental design method, Taguchi, is applied to generate new offspring.

3 **EDA/L.** The hybridisation of estimation of distribution algorithm with local search algorithms (EDA/L; Zhang et al., 2003), in which a cheap local search algorithm is applied to all newly generated individuals, and an expensive local search algorithm is applied only to the best individuals at each generation.

4 **LEA.** The evolutionary algorithm based on level-set evolution and Latin squares (LEA; Wang and Dang, 2007), in which the Latin squares technique is applied to design a new and effective crossover operator, and the level set is gradually evolved to approach the global optimum.

5 **MEDA.** The multiple restart EDA (MEDA), where the described EDA in Section 3.4 is restarted multiple times.

6 **MNEW.** The multiple restart NEWUOA (MNEW), where NEWUOA is restarted multiple times.

7 **DNEW.** The multiple restart NEWUOA with the diversification scheme (DNEW) as described in Section 3.4.2.

Table 7: The comparison between WMMA and IMMA to study the performance of the decoupled hybridisation strategy on the functions with 30 dimensions. The last two columns list the p value of the rank sum test on NFEs (p_{nw}) and the best error values (p_{vw}). — denotes that the rank sum test is not necessary either because both algorithms are fully successful (no p_{vw} is available) or fully failed (no p_{nw} is available), *denotes that the algorithm stops at the maximum NFEs, that is, 3.0×10^5.

	WMMA		IMMA			
Function	$\text{AVG}_E \pm SD_E$	$\text{AVG}_N \pm SD_N$(SUC)	$\text{AVG}_E \pm SD_E$	$\text{AVG}_N \pm SD_N$(SUC)	p_{vw}	p_{nw}
f_1	0.00e+00 ± 0.00e+00	**6.09e+02** ± 2.89e+00(25)	0.00e+00 ± 0.00e+00	6.55e+03 ± 6.27e+02(25)	—	**<.001**
f_2	0.00e+00 ± 0.00e+00	**5.96e+03** ± 1.83e+02(25)	6.22e–12 ± 2.57e–12	1.01e+04 ± 1.03e+03(25)	—	**<.001**
f_3	0.00e+00 ± 0.00e+00	3.65e+04 ± 3.38e+03(25)	1.51e–08 ± 1.54e–08	**3.13e+04** ± 2.57e+03(25)	—	**<.001**
f_{28}	1.94e–34 ± 2.85e–34	**3.54e+03** ± 2.66e+02(25)	1.94e–34 ± 2.85e–34	6.30e+03 ± 3.56e+02(25)	—	**<.001**
f_{29}	4.55e–11 ± 5.44e–12	2.51e+04 ± 2.07e+04(25)	**2.61e–14** ± 7.80e–15	2.67e+04 ± 6.21e+03(25)	—	.068
f_{30}	7.21e–13 ± 4.83e–13	**2.25e+03** ± 1.28e+02(25)	1.63e–13 ± 3.26e–13	7.68e+03 ± 3.28e+03(25)	—	**<.001**
f_{31}	1.44e–11 ± 4.02e–11	5.41e+04 ± 1.77e+03(25)	3.69e–11 ± 1.10e–10	**4.69e+04** ± 1.46e+04(25)	—	**<.001**
f_{32}	3.82e–04 ± 8.21e–13	3.60e+04 ± 2.28e+04(25)	3.82e–04 ± 7.00e–13	**1.47e+04** ± 3.74e+03(25)	—	**<.001**
f_{33}	5.63e–14 ± 6.11e–14	1.06e+05 ± 8.55e+04(25)	3.09e–14 ± 3.02e–14	**2.02e+04** ± 9.15e+03(25)	—	**<.001**
f_{37}	4.44e–19 ± 6.31e–19	**4.30e+03** ± 1.73e+03(25)	4.73e–11 ± 4.06e–11	1.93e+04 ± 1.65e+04(25)	—	**<.001**
f_4	2.39e+06 ± 3.22e+05	*	**1.57e+05** ± 2.10e+04	*	**<.001**	—
f_5	6.15e+04 ± 1.38e+01	*	**1.68e+03** ± 3.27e+02	*	**<.001**	—
f_8	2.00e+01 ± 6.49e–15	*	2.00e+01 ± 1.12e–10	*	1.000	—
f_{26}	3.20e–01 ± 4.08e–02	*	**2.84e–01** ± 4.73e–02	*	**<.001**	—
f_{27}	5.09e+02 ± 5.88e+01	*	**4.13e+01** ± 4.97e+01	*	**<.001**	—
f_6	1.00e+05 ± 0.00e+00	*	**3.18e–10** ± 2.56e–10	3.11e+04 ± 3.03e+04(25)	**<.001**	**<.001**
f_7	1.58e–03 ± 3.25e–03	*	**1.31e–12** ± 4.79e–12	1.02e+04 ± 6.28e+03(25)	**<.001**	**<.001**
f_9	1.60e–03 ± 1.30e–03	5.99e+04 ± 2.52e+04(3)	**1.09e–03** ± 3.03e–03	**3.94e+04** ± 1.46e+04(25)	**<.001**	**<.001**
f_{10}	5.17e+01 ± 1.01e+01	*	**1.90e–03** ± 3.48e–03	**4.20e+04** ± 1.60e+04(25)	**<.001**	**<.001**
f_{34}	3.92e–03 ± 6.14e–03	2.34e+05 ± 1.05e+05(5)	**5.56e–13** ± 4.21e–13	**4.02e+04** ± 2.72e+04(25)	**.030**	**<.001**

8 **DEahcSPX.** The differential evolution (DE) with adaptive hill-climbing XLS (AHCXLS), called DEahcSPX (Noman and Iba, 2008).

9 **IPOP-CMA-ES.** The restart covariance matrix adaptive evolution strategy (CMA-ES) with increasing population size (IPOP-CMA-ES; Auger and Hansen, 2005).

10 **APrMF.** The adaptive probabilistic memetic framework (APrMF; Nguyen et al., 2009).

The comparisons among the well-known EAs are divided into two parts. The first part involves the comparison between IMMA and the first seven algorithms on f_{28} through f_{40}. The second part involves IMMA and the last three algorithms on f_1 through f_{35}. The division is due to the fact that we do not have the experimental results of Algorithms OGA/Q, HTGA, EDA/L, and LEA on the CEC 2005 test functions.

4.4.1 Comparison Between IMMA and the First Seven Algorithms

In the comparison between the first seven algorithms and IMMA, the test functions are grouped into two categories w.r.t. the number of dimensions. The dimension of functions in Category I (f_{28} through f_{36}) is 30, while that of Category II (f_{37} through f_{40}) is 100.

In the experiments, the algorithmic parameters of IMMA are set as follows: $M = 2n$, $T = 5$ and $L = 3$. In the comparison study, the evaluation criteria are different from those used in Section 4.1. Each of the compared algorithms uses the termination criteria as originally proposed by their authors. Since IMMA is essentially a multi-restart algorithm, the algorithm can be terminated after any cycle for which the global best solution is satisfactory. To carry out a fair comparison, in our implementation, for Category I functions, IMMA will terminate if the best solution found is no worse than OGA/Q (this criterion was also used in Tsai et al., 2004). For the Category II test problems, we terminate IMMA when the best solution found is no worse than the average best solution found by the other compared algorithms. Moreover, we terminate the algorithm when the NFE exceeds $n \times 10^5$.

Table 8 summarises the experimental results of IMMA and the results of other algorithms on the test suite. We have taken the experimental results of the compared algorithms on these problems from corresponding papers. Note that there are many other algorithms, such as the hybrid cooperative particle swarm optimisation (Bergh and Engelbrecht, 2004), the self-organizing hierarchical particle swarm optimisation with time-varying acceleration coefficients (Ratnaweera et al., 2004), the fast evolutionary programming with Cauchy mutation (Yao and Liu, 1999), and so on. We do not include the results of these algorithms since they are not as competitive as LEA (as claimed in Wang and Dang, 2007).

In the tables, NFE is the average number of fitness evaluations within 25 runs. In column p_z, the p values between the best function values obtained by IMMA and the other algorithms are shown. Here the Z-test is applied between IMMA and the first four algorithms, while the rank sum test is applied between IMMA and algorithms MEDA, MNEW, and DNEW.

From Table 8, it can be seen that IMMA performs significantly better than algorithms MEDA, MNEW, and DNEW on all functions except for f_{28} and f_{30}, which are unimodal functions. This observation shows that IMMA successfully takes advantage of EDA and NEWUOA. Moreover, it can be seen that DNEW performs at least better than

Table 8: Experimental results of the compared algorithms on the selected functions. Column p_z shows the p values obtained from the Z-test employed between IMMA and OGA/Q, HTGA, and LEA in terms of the best error values, and the p values from the rank sum test between IMMA and MEDA, MNEW, and DNEW.

Function	Algorithms	NFE	$AVG_E \pm SD_E$	p_z	Function	NFE	$AVG_E \pm SD_E$	p_z
f_{28}	IMMA	6.30e+03	0.00e+00±0.00e+00		f_{29}	9.48e+03	4.02e–17±3.00e–05	
	OGA/Q	1.13e+05	0.00e+00±0.00e+00	—		1.12e+05	4.44e–16±3.99e–17	.872
	HTGA	2.08e+04	0.00e+00±0.00e+00	—		1.66e+04	0.00e+00±0.00e+00	**<.001**
	EDA/L	—	—	—		1.06e+04	4.14e–16	—
	LEA	1.11e+05	4.73e–16±6.23e–17	**<.001**		1.06e+05	3.27e–16±3.00e–17	.581
	MEDA	3.00e+05	1.52e–02±7.38e–02	**<.001**		3.00e+05	2.40e–03±4.40e–03	**<.001**
	MNEW	1.31e+02	3.80e–32±6.20e–32	**<.001**		1.23e+04	4.22e–08±5.93e–03	**<.001**
	DNEW	1.31e+02	3.80e–32±6.20e–32	**<.001**		8.49e+04	3.12e–09±2.11e–10	**<.001**
f_{30}	IMMA	7.68e+03	0.00e+00±0.00e+00		f_{31}	4.69e+04	0.00e+00±0.00e+00	
	OGA/Q	1.34e+05	0.00e+00±0.00e+00	—		2.25e+05	0.00e+00±0.00e+00	—
	HTGA	2.10e+04	0.00e+00±0.00e+00	—		1.63e+04	0.00e+00±0.00e+00	—
	EDA/L	7.91e+04	0.00e+00	—		7.50e+04	0.00e+00±0.00e+00	—
	LEA	1.30e+05	6.10e–16±2.50e–17	**<.001**		2.24e+05	2.10e–18±3.36e–18	**<.001**
	MEDA	3.00e+05	1.37e–02±2.04e–02	**<.001**		3.00e+05	4.60e–03±1.16e–02	**<.001**
	MNEW	1.36e+03	3.62e–19±2.49e–19	**<.001**		3.00e+05	2.80e+01±4.18e+01	**<.001**
	DNEW	1.36e+03	3.62e–19±2.49e–19	**<.001**		3.00e+05	7.85e+0±1.08e+01	**<.001**
f_{32}	IMMA	1.47e+04	3.82e–04±7.00e–13		f_{33}	2.02e+04	3.09e–14±3.02e–14	
	OGA/Q	3.02e+05	3.00e–02±6.45e–04	**<.001**		1.34e+05	6.02e–06±1.16e–06	**<.001**
	HTGA	1.63e+05	2.00e–02±0.00e+00	**<.001**		6.66e+04	1.00e–06±0.00e+00	**<.001**
	EDA/L	5.22e+04	3.68e–04	—		8.99e+04	3.65e–21	—
	LEA	2.87e+05	2.00e–02±4.83e–01	**<.001**		1.33e+05	2.48e–06±2.28e–06	**<.001**
	MEDA	1.70e+05	3.37e–02±2.49e–02	**<.001**		2.49e+05	1.44e–05±1.76e–05	**<.001**
	MNEW	3.00e+05	4.26e+03±4.51e+02	**<.001**		9.82e+04	3.43e–12±3.81e–12	**<.001**
	DNEW	3.00e+05	3.60e+03±3.86e+02	**<.001**		8.34e+04	2.43e–12±2.37e–12	**<.001**

Table 8: Continued

Function	Algorithms	NFE	$AVG_E \pm SD_E$	p_z	Function	NFE	$AVG_E \pm SD_E$	p_z
f_{34}	IMMA	3.64e+04	6.61e–13±6.00e–06		f_{35}	4.01e+04	0.00e+00±0.00e+00	
	OGA/Q	1.34e+05	1.87e–04±2.62e–05	<**.001**		1.13e+05	0.00e+00±0.00e+00	—
	HTGA	5.90e+04	1.00e–04±0.00e+00	<**.001**		1.43e+04	0.00e+00±0.00e+00	—
	EDA/L	1.15e+05	3.48e–21	—		—	—	—
	LEA	1.30e+05	1.73e–04±1.21e–04	<**.001**		1.10e+05	4.25e–19±4.24e–19	<**.001**
	MEDA	2.57e+05	1.31e–03±3.74e–03	<**.001**		3.00e+05	4.66e+04±1.01e+05	<**.001**
	MNEW	3.00e+05	3.55e–01±3.03e–01	<**.001**		3.00e+05	2.15e+09±0.00e+00	<**.001**
	DNEW	3.00e+05	4.34e–02±4.62e–01	<**.001**		3.00e+05	2.15e+09±0.00e+00	<**.001**
f_{36}	IMMA	1.21e+04	0.00e+00±0.00e+00		f_{37}	5.53e+04	2.77e–34±3.09e–34	
	OGA/Q	1.13e+05	0.00e+00±0.00e+00	—		1.68e+05	7.50e–01±1.10e–01	<**.001**
	HTGA	2.65e+04	0.00e+00±0.00e+00	—		6.07e+04	7.00e–01±0.00e+00	<**.001**
	EDA/L	—	—	—		1.28e+05	4.32e–03	—
	LEA	1.11e+05	6.78e–18±5.43e–18	<**.001**		1.69e+05	5.60e–01±1.10e–01	<**.001**
f_{38}	IMMA	1.29e+05	0.00e+00±0.00e+00		f_{39}	6.39e+04	3.93e+00±3.46e–10	
	OGA/Q	1.13e+05	0.00e+00±0.00e+00	—		3.03e+05	6.44e+00±2.60e–02	<**.001**
	HTGA	2.13e+04	0.00e+00±0.00e+00	—		2.66e+05	6.34e+00±0.00e+00	<**.001**
	EDA/L	—	—	—		1.70e+05	4.90e+00	—
	LEA	1.11e+05	2.68e–16±6.26e–17	<**.001**		2.90e+05	6.26e+00±2.30e–02	<**.001**
f_{40}	IMMA	1.59e+04	2.30e–03±2.70e–02					
	OGA/Q	2.46e+05	3.20e–02±6.29e–03	<**.001**				
	HTGA	1.87e+05	3.23e–02±0.00e+00	<**.001**				
	EDA/L	1.53e+05	3.20e–02	—				
	LEA	2.44e+05	2.23e–02±6.13e–03	**.001**				

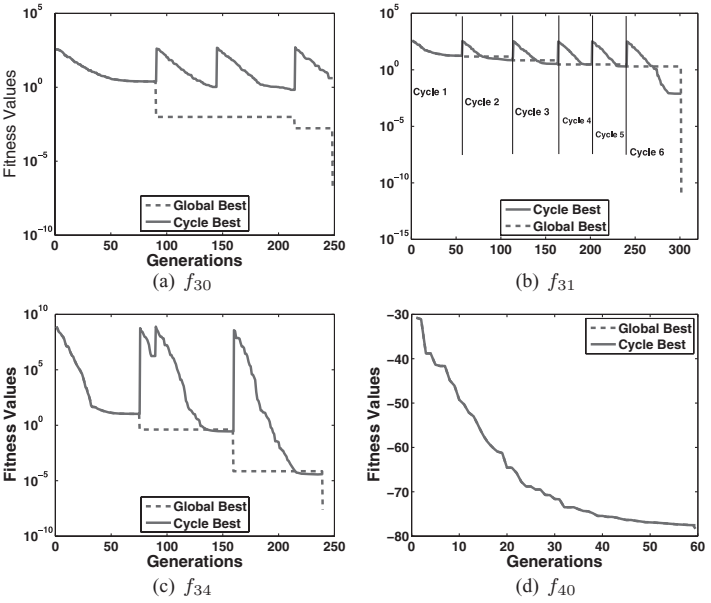


Figure 6: Typical evolution procedures of the proposed algorithm on $f_{30,31,34,40}$. Particularly in (b), we see that in six cycles, IMMA reaches the global optimum.

MNEW. Recall that DNEW and MNEW are the same except that the restart scheme in Section 3.4.2 is adopted in DNEW. This observation justifies the usefulness of the applied restart scheme.

Comparing IMMA with LEA, we see that IMMA performs significantly better than LEA on almost all the functions in terms of best error values except f_{29}. In terms of NFE, IMMA requires less than LEA except f_{38}. Comparing the results of IMMA with EDA/L, we see that EDA/L needs more NFEs on all functions to obtain similar accuracy, except for functions $f_{33,34}$ where EDA/L obtains better accuracies.

In the comparison between IMMA and HTGA, it can be seen that on average, HTGA needs fewer NFEs to reach the global optimum than IMMA on four functions, that is, $f_{29,31,35,38}$, while IMMA needs fewer NFEs on the rest of the nine functions. Notice that except for functions $f_{29,31,35,38}$, the remaining functions have high epistasis properties. This implies that IMMA has a better capability in dealing with epistasis than that of HTGA. The observation that IMMA obtains significantly better solutions than those of HGTA when solving $f_{37,39,40}$, which are of high dimensions and high epitasis, also confirms the implication. Comparing IMMA with OGA/Q, we see that IMMA requires much fewer NFEs on all the selected functions except f_{38}, but the obtained objective function values are no worse than OGA/Q.

Figure 6 shows typical evolution procedures of the proposed algorithm on $f_{30,31,34}$ and f_{40}. In the figures, global best (the dotted line) shows the evolution of the best solution found so far, while cycle best (the dashed line) displays the evolution of the best solutions in the cycles. In Figure 6(a), we see that the developed algorithm needs four cycles to reach the global optimum. The figure demonstrates that in some problem instances (e.g., f_{40}), IMMA can find the global optimum in one cycle, while in most optimisation problems, several cycles are required to reach the attraction basin of the global optimum. Figure 6 shows that IMMA can approach the attraction basin

gradually through learning from previous search. For example, in Figures 6(b) and 6(c), the searches in later cycles find better solutions gradually until the attraction of the global optimum has been reached. While in Figure 6(a), the later cycles cannot always find better solutions than previous cycles, but the algorithm finally adapts to reach the global attraction basin. More interestingly, in Figure 6(a), the global optimum is obtained by applying the local search algorithm from a worse initial point.

In summary, we can claim that the novel hybrid strategy can form the basis of efficient memetic algorithms. Moreover, the empirical study indicates that on average, IMMA is comparable with HTGA if not better, and is significantly better than the other compared algorithms on these functions. Note that while these results are valid for 30-dimensional functions, they may not generalise to other dimensions.

4.4.2 The Comparison Among IMMA, DEahcSPX and IPOP-CMA-ES

Only two algorithms, that is, DEahcSPX (Noman and Iba, 2008) and IPOP-CMA-ES (Auger and Hansen, 2005), have been applied to the test functions used in CEC 2005. The dimensions of the test functions adopted in the CEC 2005 competition, that is, the first 25 test functions as listed in Appendix A, are restricted to 10, 30, and 50. The algorithmic parameters of IMMA are set the same as in Section 4.4.1.

DEahcSPX (Noman and Iba, 2008) applies an adaptive crossover local search, called AHCXLS, to improve the best found solution at each generation. This hybrid strategy is similar to that developed in Zhang et al. (2003) except that the crossover LS rather than the classical LS is applied. IPOP-CMA-ES can be considered as a multi-restart version of CMA-ES in Hansen and Kern (2004). Several stop criteria have been designed to restart CMA-ES. In each restart, the population size is increased by a factor of two over the previous population size.

We summarise the experimental results in Figure 7, where the bar charts show the average values of 25 runs. In the plots, the average NFEs (AVG_N) are shown if the functions can be solved in all 25 trials, while the average best error values (AVG_E) are shown if the functions cannot be solved in all 25 trials. The p values shown are the rank sum test over the corresponding experimental results. Detailed experimental results can be found in the online appendix.[1]

In the comparison between IMMA and DEahcSPX in terms of NFEs, we see that IMMA uses significantly fewer NFEs than DEahcSPX on f_1 through f_3 and f_6. Also, it can be seen that IMMA performs better than DEahcSPX on f_7 through f_{10} but worse on f_4 and f_5 in terms of the best error values. From Table A1 in the appendix, we can see that f_4 is noisy and f_5 is discontinuous. Thus, they may break the POP assumptions of the developed algorithm. Also, these properties may make NEWUOA employed in IMMA less effective. In summary, we may conclude that IMMA works better than DEahcSPX for test problems that are continuous and without noise.

In the comparison between IMMA and IPOP-CMA-ES on these functions that can be successfully solved in terms of NFEs, we observe that IMMA performs worse than IPOP-CMA-ES only on $f_1(n = 30, 50)$ and $f_3(n = 50)$. In the comparison between IMMA and IPOP-CMA-ES on the remaining functions (in total, 21 functions) in terms of the best error values, in the case where $n = 10$, we see that IMMA performs better than IPOP-CMA-ES in 10 functions, worse in eight, and comparable in three. In the case where $n = 30$, IMMA is better than IPOP-CMA-ES in 11 functions, worse in seven, and

[1] http://www.cpib.ac.uk/downloads/onlinesupplementary.pdf

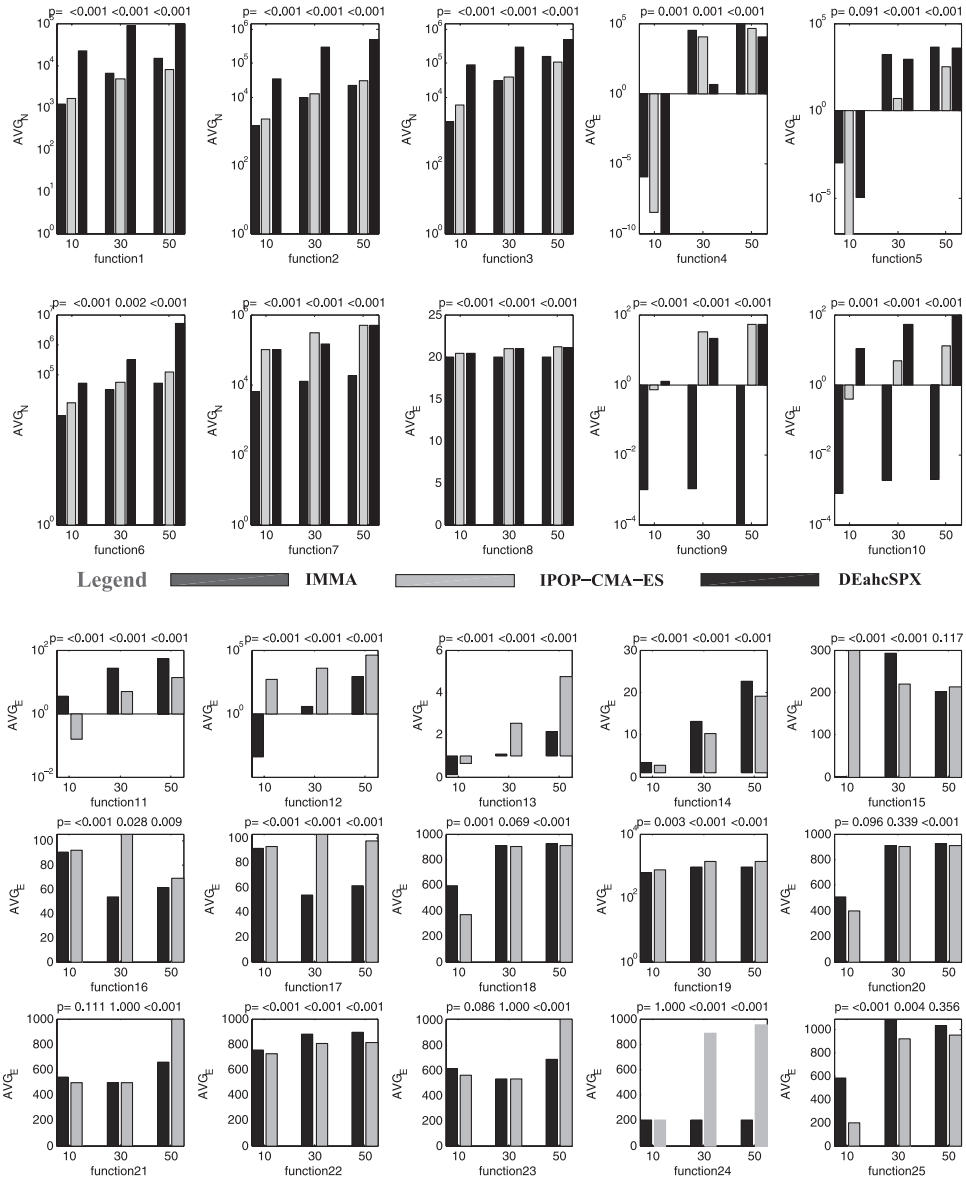


Figure 7: Comparative results of IMMA on the CEC 2005 test problems with DEahcSPX and IPOP-CMA-ES. The x axis shows the dimension, and the y axis is either the average function error value (AVG_E) or the average number of fitness evaluations (AVG_N). The p values shown in the plots are the p values obtained through the rank-sum test between IMMA and IPOP-CMA-ES.

no different in three, while in the case where $n = 50$, IMMA achieves better results in 11 functions, worse in six, and no different in four.

Note that the worst performance of IMMA occurs when solving the functions $f_{4,5,11,14,18,22,25}$. From Table A1 in the appendix, we see that most of these functions are with high-epistasis, very high-multimodal, and medium discontinuity and noise. This

Table 9: The comparison between IMMA and APrMF on the functions that can be solved successfully in terms of NFEs.

Function(dim)	Algorithms	NFEs	Function(dim)	NFEs
$f_1(10)$	IMMA	1.20e+03	$f_1(30)$	6.60e+03
	APrMF	**6.00e+02**		**6.00e+02**
$f_2(10)$	IMMA	**1.50e+03**	$f_2(30)$	**1.00e+04**
	APrMF	1.10e+04		8.70e+04
$f_3(10)$	IMMA	2.00e+03	$f_3(30)$	3.13e+04
	APrMF	**6.00e+02**		**1.00e+03**
$f_6(10)$	IMMA	**4.20e+03**	$f_7(30)$	**1.01e+04**
	APrMF	1.05e+04		2.55e+04
$f_9(10)$	IMMA	6.90e+03	$f_9(30)$	3.94e+04
	APrMF	**1.00e+03**		**5.90e+03**

confirms our observation that IMMA cannot perform well on functions with noise and discontinuity. Moreover, it can be seen that in the case where $n = 10$, IMMA does not perform as well as IPOP-CMA-ES on $f_{18,20,22,24}$, which are hybrid composition functions, while along the increase of the function dimensions, we see that the performance of IMMA becomes better. These observations indicate that IMMA may perform better for high-dimensional functions.

Tables 9 and 10 show the results in comparison with APrMF (Nguyen et al., 2009). Table 9 lists the results for those functions that can be solved successfully. The results show that IMMA needs fewer NFEs than APrMF on $f_{2,6,7}$ which have high epistasis, while APrMF is good on $f_{1,3,9}$ which have no epistasis. On the other hand, for these functions that are not easy to be solved as seen in Table 10, IMMA performs significantly better than APrMF on almost all the functions except for $f_{14,16,19}$ according to the Z-test between the algorithms.

In summary, we can conclude that on average IMMA can perform significantly better than DEahcSPX and APrMF on almost all functions. This justifies the following two conclusions: (1) the developed novel hybridisation strategy works better than the coupled hybridisation strategy as used in APrMF; (2) the incorporation of classical LS improves the algorithmic efficiency. On the other hand, we see that IPOP-CMA-ES performs better than IMMA on those functions with noise and discontinuous properties when $n = 10$. However, IMMA performance improves along with the dimension increase of these functions. On average, we can conclude that IMMA is comparable with IPOP-CMA-ES on the CEC2005 test functions.

4.5 Scalability

In this section, we carry out experiments to investigate the effect of dimensionality on the performance of IMMA. We use the same test functions as in Noman and Iba (2008), including f_{26} through f_{34} and f_{37} to carry out the analysis. We carry out the study on these functions with dimensions at $n = 10$, 50, 100, and 200. The criteria described in Section 4.1 are applied. The algorithmic parameters are set as $M = 2n$, and $L = 3$.

Table 11 lists the results, including the best error values and the number of fitness evaluations, obtained by IMMA and DEahcSPX which are copied from Noman and Iba (2008). The global optimal values of these functions with different dimensions are all zero. From the table, it is clear that IMMA performs significantly better than DEahcSPX on all functions except for f_{26} ($n = 10, 50, 200$) in terms of the best error values and NFEs except for $f_{29}(n = 10)$. Moreover, we can see that IMMA reaches the attraction basin of

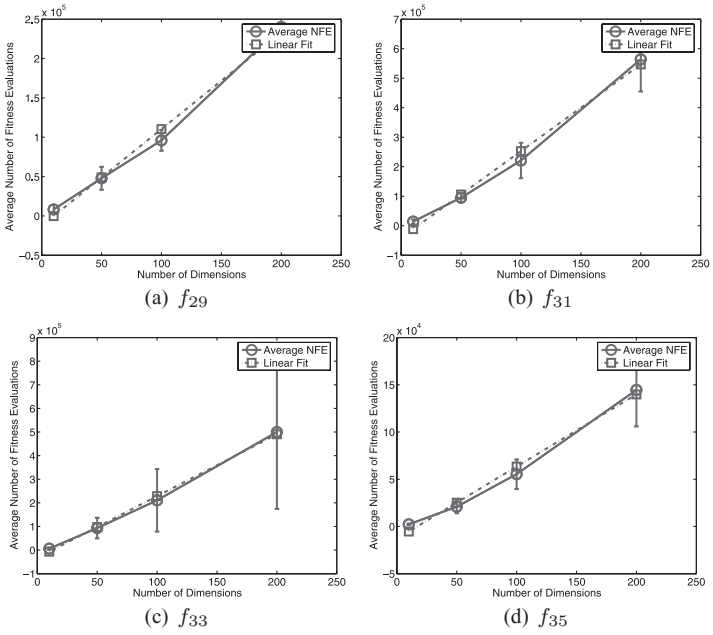
Table 10: The comparison between IMMA and APrMF on the functions that cannot be solved successfully, where p_a is the value obtained from the Z-test and the values in parentheses are the number of successful trials.

Function(dim)	Algorithms	$AVG_E \pm SD_E$	p_a	Function(dim)	$AVG_E \pm SD_E$	p_a
f_4(10)	IMMA	**1.05e–06**±1.39e–06(17)		f_4(30)	**2.94e+04**±6.16e+03	
	APrMF	1.48e+03±1.01e+03(0)	**0.000**		3.61e+04±6.63e+03	**.001**
f_5(10)	IMMA	**1.08e–03**±3.07e–03		f_5(30)	**1.68e+03**±3.27e+02	
	APrMF	2.23e+02±2.58e+02	<**0.001**		6.66e+03±1.28e+03	<**.001**
f_7(10)	IMMA	**4.34e–15**±4.92e–15(25)		f_8(10)	**2.00e+01**±3.43e–13	
	APrMF	8.07e–03±1.14e–02(20)	**0.002**		2.02e+01±1.05e–01	<**.001**
f_8(30)	IMMA	**2.00e+01**±1.12e–10		f_{10}(10)	**7.89e–04**±2.53e–03	
	APrMF	2.06e+01±1.91e–01	<**0.001**		2.30e+01±6.35e+0	<**.001**
f_{10}(30)	IMMA	**1.90e–03**±3.48e–03		f_{11}(10)	**3.59e+00**±8.39e–01	
	APrMF	2.17e+02±3.36e+01	<**0.001**		5.70e+00±1.01e+00	<**.001**
f_{11}(30)	IMMA	**2.59e+01**±1.99e+00		f_{12}(10)	**4.00e–04**±2.00e–03(25)	
	APrMF	2.93e+01±2.54e+00	0.129		2.38e–02±7.84e–02(19)	<**.001**
f_{12}(30)	IMMA	**3.77e+00**±8.93e+00(16)		f_{13}(10)	**1.32e–01**±1.04e–01 (6)	
	APrMF	4.42e+02±6.72e+02(0)	**0.003**		7.07e–01±2.07e–01 (0)	**0.010**
f_{13}(30)	IMMA	**1.08e+00**±2.41e–01		f_{14}(10)	3.43e+00±2.74e–01 (0)	
	APrMF	7.72e+00±2.26e+00	<**0.001**		**4.99e–02**±2.66e–02(19)	<**.001**
f_{14}(30)	IMMA	1.32e+01±2.96e–01		f_{16}(10)	**9.11e+01**±1.93e+01	
	APrMF	**5.41e–01**±1.88e–01	<**0.001**		1.37e+02±1.18e+01	<**.001**
f_{16}(30)	IMMA	5.40e+01±5.65e+00		f_{19}(10)	**6.16e+02**±2.20e+02	
	APrMF	**1.12e+00**±5.12e+00	<**0.001**		7.21e+02±1.20e+02	0.178
f_{19}(30)	IMMA	9.06e+02±1.27e+0				
	APrMF	**5.34e+02**±4.31e+01	**0.025**			

Table 11: The scalability analysis. Column $\text{AVG}_E \pm SD_E$ shows the best error value criterion, and $\text{AVG}_N \pm SD_N$ shows the NFE criterion. — in the table means that the algorithms cannot reach the global optimum within the maximum NFEs, NA denotes that DEahcSPX results are not available. p_s shows the Z-test results in terms of the best error values and NFEs.

Function	Dimension	IMMA		DEahcSPX		p_s
		$\text{AVG}_E \pm SD_E$	$\text{AVG}_N \pm SD_N$	$\text{AVG}_E \pm SD_E$	$\text{AVG}_N \pm SD_N$	
f_{26}	10	1.40e–01 ± 5.00e–02	—	**9.98e–02**±3.47e–08	—	**.001**
	50	4.12e–01 ± 4.40e–02	—	**4.00e–01**±1.00e–01	—	.588
	100	**6.76e–01** ± 5.97e–02	—	3.11e+00±5.79e–01	—	**<.001**
	200	1.15e+00 ± 1.12e–01	—	**1.10e+01**±4.38e–01	—	.871
f_{27}	10	**7.27e+00** ± 6.41e+00	—	1.80e+01±1.31e+01	—	**.001**
	50	**2.61e+02** ± 1.40e+02	—	1.41e+03±2.90e+02	—	**<.001**
	100	**1.78e+03** ± 7.80e+02	—	4.06e+10±6.57e+10	—	**.005**
	200	**5.89e+04** ± 9.90e+03	—	4.21e+13±1.74e+13	—	**<.001**
f_{28}	10	2.31e–36 ± 5.48e–36	**1.18e+03** ±9.64e+01	**1.81e–38**±4.94e–38	2.29e+04±1.30e+03	**<.001**
	50	**8.06e–34** ± 1.09e–33	1.39e+04 ±6.84e+02	8.80e–09±2.80e–08	NA	**<.001**
	100	**1.58e–33** ± 1.55e–33	4.02e+04 ±1.51e+03	5.01e+01±8.94e+01	—	**<.001**
	200	**7.44e–33** ± 9.07e–33	1.15e+05 ±4.54e+04	7.01e+03±1.07e+03	—	**<.001**
f_{29}	10	8.10e–15 ± 4.29e–15	8.22e+04 ±5.64e+03	**2.66e–15**±0.00e+00	**3.64e+04**±1.76e+04	**<.001**
	50	**5.23e–14** ± 1.55e–14	4.78e+04 ±1.47e+04	1.69e–05±8.86e–06	NA	**<.001**
	100	**7.69e–14** ± 1.22e–14	9.60e+04 ±1.33e+04	1.91e+00±3.44e–01	—	**<.001**
	200	**9.00e–14** ± 2.57e–14	2.40e+05 ±4.27e+04	8.45e+00±4.13e–01	—	**<.001**
f_{30}	10	**0.00e+00** ± 0.00e+00	1.66E+04 ±1.13e+03	4.77e–02±2.55e–02	—	**<.001**
	50	**1.40e–18** ± 1.05e–18	1.91e+04 ± 5.23e+03	2.96e–03±5.64e–03	—	**<.001**
	100	**4.58e–18** ± 2.07e–18	5.12e+04 ± 1.21e+04	1.23e+00±2.14e–01	—	**<.001**
	200	**1.63e–17** ± 6.59e–18	1.32e+05 ± 5.34e+03	6.08e+01±9.30e+00	—	**<.001**
f_{31}	10	**0.00e+00** ± 0.00e+00	**1.42e+04** ± 7.35e+03	1.60e+00±1.61e+00	8.43e+04±2.20e+04	**<.001**
	50	**5.00e–17** ± 9.17e–17	9.50e+04 ± 1.50e+04	3.47e+01±9.23e+00	—	**<.001**
	100	**2.60e–16** ± 5.41e–16	2.21e+05 ± 5.99e+04	4.75e+02±6.55e+01	—	**<.001**
	200	**1.01e–15** ± 2.15e–15	5.64e+05 ± 1.09e+05	1.53e+03±8.31e+01	—	**<.001**

f_{32}	10	**1.27e−04** ± 1.10e−13	8.97e+03 ± 3.96e+03	4.74e+00±2.37e+01	—	**<.001**
	50	**6.36e−04** ± 2.57e−12	3.80e+04 ± 1.17e+04	9.56e+02±2.88e+02	—	**<.001**
	100	**1.27e−03** ± 1.75e−12	—	2.48e+04±2.17e+03	—	**<.001**
	200	**2.55e−03** ± 1.57e−11	—	6.61e+04±1.44e+03	—	**<.001**
f_{33}	10	6.41e−19 ± 1.02e−19	**7.68e+03** ± 7.31e+02	**4.71e−32**±1.12e−47	2.05e+04±1.16e+03	**<.001**
	50	**9.24e−19** ± 1.19e−18	3.39e+04 ± 2.53e+04	2.49e−03±1.24e−02	—	**<.001**
	100	**5.81e−18** ± 4.11e−18	6.89e+04 ± 6.04e+04	4.34e+00±1.75e+00	—	**<.001**
	200	**7.48e−18** ± 6.63e−18	1.33e+05 ± 2.68e+04	2.27e+01±5.73e+00	—	**<.001**
f_{34}	10	1.35e−32 ± 1.99e−39	**6.01e+03** ± 5.08e+03	1.35e−32±5.59e−48	2.16e+04±1.29e+03	**<.001**
	50	**1.35e−32** ± 1.98e−37	9.31e+04 ± 4.36e+04	2.64e−03±4.79e−03	—	**<.001**
	100	**3.55e−32** ± 1.78e−31	2.11e+05 ± 1.32e+05	7.25e+01±2.44e+01	—	**<.001**
	200	**4.29e−32** ± 1.03e−31	4.99e+05 ± 3.25e+05	6.24e+04±4.77e+04	—	**<.001**
f_{37}	10	**3.11e−37** ± 1.61e−37	**2.27e+03** ± 9.02e+02	3.19e−01±1.10e+00	5.29e+04±1.50e+04	**<.001**
	50	**5.28e−35** ± 4.03e−35	2.09e+04 ± 6.86e+03	1.63e+02±3.02e+02	—	**<.001**
	100	**2.77e−34** ± 3.09e−34	5.53e+04 ± 1.57e+04	1.45e+05±1.11e+05	—	**<.001**
	200	**2.93e−33** ± 7.06e−33	1.44e+05 ± 3.86e+04	1.11e+08±2.63e+07	—	**<.001**



(a) f_{29} (b) f_{31}

(c) f_{33} (d) f_{35}

Figure 8: The number of fitness evaluations versus the number of dimensions on $f_{29,31,33,35}$.

the global optimum linearly with respect to the dimensionality, in cases for which the accuracy level is achieved.

Figure 8 shows the average number of fitness evaluations against the dimensionality on $f_{29,31,33,35}$. The dimensionality is on the x axis, and the y axis shows the average number of fitness evaluations used to reach the predefined precision in 25 runs. Error bars indicate the standard deviation of the number of fitness evaluations. The solid line is the straight line fit between the average number of fitness evaluations and the dimensionality. From Figure 8, the number of fitness evaluations exhibits an approximately linear relationship w.r.t. the number of dimensions. We can also see that the standard deviations increase along with the dimensionality.

4.6 Additional Materials

Following the release of the benchmark problems in the Genetic and Evolutionary Computation Conference (GECCO) workshop for real-parameter optimisation in 2009 (Hansen et al., 2009), we applied the developed algorithm to these problems. The experimental results including the details of the execution procedure and the comparison with the winner of the competition, called BIPOP (Hansen et al., 2010) are available elsewhere.[2]

To give a brief impression of the results, we include Figure 9, which summarises the comparison of IMMA and BIPOP on the 24 benchmark problems with various dimensions, using the format specified in the GECCO 2009 competition. In Figure 9, ERT stands for the expected run time to surpass $f_{\text{opt}}+\Delta f$, where Δf is a given target

[2]http://www.cpib.ac.uk/downloads/onlinesupplementary.pdf

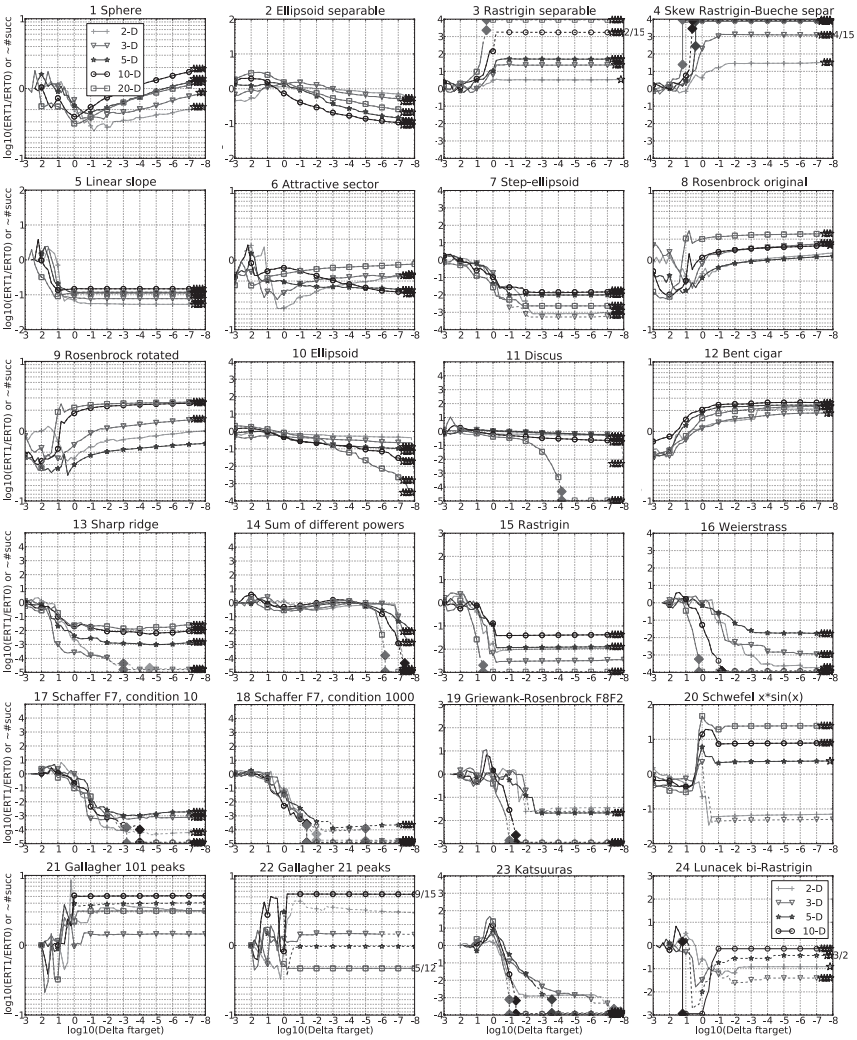


Figure 9: Ratio of ERT for IMMA over BIPOP versus $\log_{10}(\Delta f)$ in 2:+, 3:◯, 5:⋆, 10:○ and 20:□ dimensions. Ratios $< 10^0$ indicate an advantage of IMMA, as smaller values are always better. For a complete description of the figures, see text.

difference to the optimal function value f_{opt}. The figure shows the log ratio of ERT for IMMA over ERT for BIPOP versus $\log_{10}(\Delta f)$ in 2, 3, 5, 10, and 20 dimensions. In Figure 9, log-ratios < 0 indicate an advantage of IMMA, as smaller values are always better. The line becomes dashed when the ERT of the compared algorithms exceeds three times the median of the trial-wise overall number of f evaluations. The best achieved Δf value for each algorithm is indicated by a filled symbol. The dashed line continues as the fraction of successful trials of the other algorithms, where 0 means 0%, and the y axis limits mean 100%, values below zero for IMMA. The line ends when no further algorithm reaches Δf. The number of successful trials is given only if it was in $\{1 \ldots 9\}$ for IMMA (1st number) and nonzero for BIPOP (2nd number). The results are significant with $p = .05$ for one star and $p = 10^{-\#\star}$ otherwise, with Bonferroni

correction within each figure. From the figure, we see that for two test problems, $f_{21,22}$, there are no significant differences for all the dimensions. For the 2D functions, it can be seen that BIPOP performs significantly better than IMMA in three cases ($f_{3,4,12}$), IMMA significantly better in sixteen ($f_{1,2}$, f_5 through f_7, $f_{10,11}$, f_{13} throuh f_{19}, and $f_{23,24}$), with no significant difference in the remaining five ($f_{8,9}$, and f_{20} through f_{22}). Similar patterns of results can be seen for the other dimensionalities of functions, with IMMA outperforming BIPOP more frequently than the reverse. From this observation, we state that overall, IMMA performs better than BIPOP on the GECCO 2009 competition functions in terms of ERT.

5 Discussion

One of the main contributions in the developed memetic algorithm framework is to isolate the EA and the local search. Under this framework, traditional principles for efficient EA design will be profoundly changed. That is, we no longer need to focus on balancing exploration and exploitation, but instead focus on improving the capabilities of effective exploration and intelligent restart. The aim of effective exploration is to find a very promising solution using the least computational cost, while the aim of intelligent restart is to develop good strategies to take advantage of the search history to improve search efficiency. Moreover, the multiple-cycle structure provides a possibility to escape from local optima, once trapped. From the component studies, we found that an IMMA with high selection pressure does not necessarily worsen the performance of the resultant algorithm. This observation implies that the traditional EA design criteria, such as the selection pressure and the transformation from exploration to exploitation (Yao and Liu, 1999), are not appropriate to guide the EA design under the proposed framework. In this case, future research should concentrate on developing EAs with strong exploration capability and on proposing intelligent restart criteria.

Apart from the guided mutation operator which can incorporate previous history information, it is desirable to develop more intelligent strategies which can take advantage of the previous search history and reduce the randomness of the evolutionary search in order to improve robustness.

Another important issue in the framework is the choice of proper local optimisers. There are a diverse and large set of local optimisers for unconstrained and constrained continuous optimisation problems. Some local optimisers are developed particularly for some kind of problems, for example, N2FB (Dennis et al., 1981), which is particularly developed for least square problems. When applying the developed algorithm to a practical problem, specialised local optimisers would be more useful than NEWUOA in some situations. Moreover, in the case that local optimisers have their own control parameters, we may need to develop some strategies to search for the optimal control parameters during the optimisation process, and also develop multi-meme strategies (Krasnogor et al., 2006) such as those successfully deployed for combinatorial optimisation problems.

6 Conclusion

In this paper, we have studied the application of a memetic algorithm to continuous optimisation problems, and proposed an intelligent multi-restart memetic framework. We developed an EDA-like algorithm based on the proposed framework. In the implementation, an adaptive multivariate model is constructed and used to sample offspring. To replace a solution, we sample more than one solution from the probability model,

and pick the best of these to replace the current one. A derivative-free optimisation algorithm, NEWUOA, is applied to improve the current best solution when the current search cannot find a better solution in some consecutive generations. When the stop criterion has not been met, we restart the search. Hence, the new search will be carried out intelligently by incorporating the history information of previous searches. Experiments on some commonly used benchmark global optimisation problems showed that the proposed algorithm is very competitive with the best known EAs, including the winner of the CEC 2005 competition, and significantly better than other EAs.

In summary, we believe that the developed algorithm, based on our new framework, achieves highly competitive results (in terms of the solution quality) across a wide range of functions of various dimensions. Often it does this by using fewer fitness evaluations than other competitive algorithms. We also believe that the framework is relatively simple and flexible. Hence, we believe that this framework is a significant contribution, worthy of future study.

Moreover, we intend to explore four interesting avenues of research. The first avenue will be to examine the use of the niching technique (De Jong, 1975) to help the search escape from local optima and hence make the diversification more intelligent. The second avenue will be to apply the developed framework to a range of problems in combinatorial optimisation. The third avenue will be to modify the framework to accommodate the search for multiple optimum solutions in multi-modal optimisation problems. The fourth avenue will be to investigate a completely adaptive intelligent multi-restart algorithm, based on the proposed framework, to save manpower in practice.

Acknowledgment

This work was supported by BBSRC and EPSRC through grants BB/D019613/1 (The Centre for Plant Integrative Biology), EP/H010432/1 and EP/J004111/1.

References

Auger, A., and Hansen, N. (2005). A restart CMA evolution strategy with increasing population size. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC2005)*, pp. 1769–1776.

Bäck, T. (1996). *Evolutionary algorithms in theory and practice*. Oxford, UK: Oxford University Press.

Bäck, T., Hammel, U., and Schwefel, H.-P. (1997). Evolutionary computation: Comments on the history and current states. *IEEE Transactions on Evolutionary Computation*, 1(1):3–17.

Bambha, N., Bhattacharyya, S., Teich, J., and Zitzler, E. (2004). Systematic integration of parameterized local search into evolutionary algorithm. *IEEE Transactions on Evolutionary Computation*, 8(2):137–155.

Bergh, F., and Engelbrecht, A. (2004). A cooperative approach to particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):225–239.

Bosman, P. A. N., and Thierens, D. (2000). Expanding from discrete to continuous estimation of distribution algorithms: The IDEA. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel (Eds.), *Parallel problem solving from nature, PPSN VI. Lecture notes in computer science*, Vol. 1917 (pp. 767–776). Berlin: Springer-Verlag.

De Jong, K. (1975). An analysis of behavior of a class of genetic adaptive systems. PhD thesis, The University of Michigan, Ann Arbor, Michigan.

Dennis, J., Jr., Gay, D., and Welsch, R. (1981). An adaptive nonlinear least-squares algorithm. *ACM Transactions on Mathematical Software*, 7(3):348–368.

Eiben, A. E., and Smith, J. E. (2003). *Introduction to evolutionary computing*. Berlin: Springer-Verlag.

Fogel, D. B. (Ed.). (1999). *Evolutionary computation: The fossil record*. New York: Wiley.

Glover, F. (1998). A template for scatter search and path relinking. In *The Third European Conference on Artificial Evolution. Lecture notes in computer science,* Vol. 1363 (pp. 13–54). Berlin: Springer-Verlag.

Glover, F., and Laguna, M. (1998). *TABU search*. Norwell, MA: Kluwer Academic Publishers.

Hansen, N., Auger, A., Ros, R., Finck, S., and Posik, P. (2010). Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009. In *Workshop Proceedings of the Genetic and Evolutionary Computation Conference (GECCO2010)*. ACM.

Hansen, N., Finck, S., Ros, R., and Auger, A. (2009). Real-parameter black-box optimization benchmarking 2009: Noiseless function definitions. (Technical Report RR-6829) INRIA, Orsay Cedex, France.

Hansen, N., and Kern, S. (2004). Evaluating the CMA evolution strategy on multimodal test functions. In *Parallel problem solving from nature, PPSN VIII, Lecture notes in computer science*, Vol. 3242 (pp. 282–291). Berlin: Springer-Verlag.

Hansen, N., and Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195.

Hart, W. (1994). *Adaptive global optimization with local search*. PhD thesis, University of California, San Diego, California.

Hart, W., Krasnogor, N., and Smith, J. E. (Eds.). (2005). *Recent advances in memetic algorithms*. *Studies in fuzziness and soft computing*, Vol. 166. Berlin: Springer-Verlag.

Houck, C., Joines, J., Kay, M., and Wilson, J. (1997). Empirical investigation of the benefits of partial lamarckianism. *Evolutionary Computation*, 5(1):31–60.

Houck, C., Joines, J., Kay, M., and Wilson, J. (2004). A study on the use of self-generation in memetic algorithms. *Natural Computing*, 3(1):53–76.

Jones, D. (1995). DIRECT. In R. Horst and P. Pardalos (Eds.), *Handbook of Global Optimization*. Norwell, MA: Kluwer Academic Publishers.

Krasnogor, N. (2009). Memetic algorithms. In G. Rozenberg, T. Back, and J. Kok (Eds.), *Handbook of natural computation*. Berlin: Springer-Verlag.

Krasnogor, N., Aragon, A., and Pacheco, J. (2006). Memetic algorithms. In *Metaheuristics in neural networks learning*. Berlin: Springer-Verlag.

Krasnogor, N., and Smith, J. (2000). A memetic algorithm with self-adaptive local search: TSP as a case study. In L. D. Whitley, D. E. Goldberg, E. Cantu-Paz, L. Spector, and H.-G. Beyer (Eds.), *Proceedings of Genetic and Evolutionary Computation Conference (GECCO2000)*, pp. 987–994.

Krasnogor, N., and Smith, J. (2005). A tutorial for competent memetic algorithms: Model, taxonomy, and design issues. *IEEE Transactions on Evolutionary Computation*, 9(5):474–488.

Larrañaga, P., and Lozano, J. A. (2002). *Estimation of distribution algorithms: A new tool for evolutionary computation*. Norwell, MA: Kluwer Academic Publishers.

Lee, C., and Yao, X. (2004). Evolutionary programming using mutations based on the Lévy probability distribution. *IEEE Transactions on Evolutionary Computation*, 8(1):1–13.

Leung, Y. W., and Wang, Y. (2001). An orthogonal genetic algorithm with quantization for global numerical optimization. *IEEE Transactions on Evolutionary Computation*, 5:41–51.

Lozano, M., Herrera, F., Krasnogor, N., and Molina, D. (2004). Real-coded memetic algorithms with crossover hill-climbing. *Evolutionary Computation*, 12(3):273–302.

Michalewicz, Z. (1996). *Genetic algorithm + Data structures = Evolution programs*. Berlin: Springer-Verlag.

Molina, D., Herrera, F., and Lozano, M. (2005). Adaptive local search parameters for real-coded memetic algorithms. In *The 2005 IEEE Congress on Evolutionary Computation*, pp. 888–895.

Molina, D., Lozano, M., and García-Martínez, C. (2010). Memetic algorithms for continuous optimization based on local search chains. *Evolutionary Computation Journal*, 18(1):27–63.

Moscato, P. (1989). On evolution, search, optimization, gas and martial arts: Toward memetic algorithms. (Technical report C3P, Vol. 826). California Institute of Technology, Caltech Concurrent Computing Program, Pasadena, California.

Mühlenbein, H., and Paaß, G. (1996). From recombination of genes to the estimation of distributions I. Binary parameters. In *Parallel problem solving from nature, PPSN IV. Lecture notes in computer science*, Vol. 1411 (pp. 178–187). Berlin: Springer-Verlag.

Nguyen, Q., Ong, Y.-S., and Krasnogor, N. (2007). A study on the design issues of memetic algorithms. In *Proceedings of the 2007 IEEE Congress on Evolutionary Computation*, pp. 2390–2397.

Nguyen, Q., Ong, Y.-S., and Lim, M. (2009). A probabilistic memetic framework. *IEEE Transactions on Evolutionary Computation*, 13(3):604–623.

Nguyen, Q., Ong, Y.-S., Lim, M., and Krasnogor, N. (2007). Adaptive cellular memetic algorithms. *Evolutionary Computation Journal*, 17(2):231–256.

Noman, N., and Iba, H. (2008). Accelerating differential evolution using an adaptive local search. *IEEE Transactions on Evolutionary Computation*, 12(1):107–125.

Ong, Y., and Keane, A. (2004). Meta-Lamarckian learning in memetic algorithms. *IEEE Transactions on Evolutionary Computation*, 8(2):99–110.

Ong, Y.-S., Lim, M., Zhu, N., and Wong, K.-W. (2006). Classification of adaptive memetic algorithms: A comparative study. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 36(1):141–152.

O'Reilly, U., and Oppacher, F. (1995). Hybridized crossover-based search techniques for program discovery. In *Proceedings of the 1995 World Conference on Evolutionary Computation*, Vol. 2, pp. 573–578.

Powell, M. (2006). *The NEWUOA software for unconstrained optimization without derivatives*. *Nonconvex optimization and its applications*, Vol. 83 (pp. 255–297). Berlin: Springer-Verlag.

Powell, M. (2008). Developments of NEWUOA for minimization without derivatives. *IMA Journal of Numerical Analysis*, 28:649–664.

Ratnaweera, A., Halgamuge, S., and Watson, H. (2004). Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Transactions on Evolutionary Computation*, 8(3):240–255.

Robert, C., and Casella, G. (2004). *Monte Carlo statistical methods*, 2nd ed. Berlin: Springer-Verlag.

Satoh, H., Yamamura, M., and Kobayashi, S. (1996). Minimal generation gap model for GAs considering both exploration and exploitation. In *Proceedings of IIZUKA'96*, pp. 494–497.

Schwefel, H. (1995). *Evolution and optimum seeking*. New York: Wiley.

Smith, J. (2003). Co-evolving memetic algorithms: A learning approach to robust scalable optimization. In *Proceedings of the 2003 Congress on Evolutionary Computation*, pp. 498–505.

Smith, J. (2007). Coevolving memetic algorithms: A review and progress report. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 37(1):6–17.

Stützle, T. (1999). Iterated local search for the quadratic assignment problem. Technical Report AIDA-99-03, Darmstadt University of Technology, Computer Science Department, Intellectics Group.

Sudholt, D. (2006). Local search in evolutionary algorithms: The impact of the local search frequency. In *Proceedings of the 17th International Symposium on Algorithms and Computation. Lecture notes in computer science*. Vol. 4288 (pp. 259–368). Berlin: Springer-Verlag.

Sudholt, D. (2009). The impact of parameterization in memetic evolutionary algorithms. *Theoretical Computer Science*, 410:2511–2528.

Suganthan, P., Hansen, N., Liang, J., Deb, K., Chen, Y.-P., Auger, A., and Tiwari, S. (2005). Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Technical report, Nanyang Technological University, Singapore.

Sun, J., Zhang, Q., and Tsang, E. (2005). DE/EDA: A new evolutionary algorithm for global optimisation. *Information Sciences*, 169(3-4):249–262.

Torn, A., and Zilinskas, A. (1989). Global optimization. *Lectures Notes in Computer Science*, 350.

Tsai, J.-T., Liu, T.-K., and Chou, J.-H. (2004). Hybrid Taguchi-genetic algorithm for global numerical optimization. *IEEE Transactions on Evolutionary Computation*, 8(4):365–377.

Tsutsui, S., and Goldberg, D. (2002). Simplex crossover and linkage identification: Single-stage evolution vs. multi-stage evolution. In *Proceedings of IEEE International Conference on Evolutionary Computation*. (Illi GAL Report 2002001).

Tsutsui, S., Pelikan, M., and Goldberg, D. (2001). Evolutionary algorithm using marginal histogram models in continuous domain. In *Proceedings of the 2001 Genetic and Evolutionary Computation Conference Workshop*, pp. 230–233.

Tsutsui, S., Yamamura, M., and Higuchi, T. (1999). Multi-parent recombination with simplex crossover in real coded genetic algorithms. In *Proceedings of Genetic Evolutionary Computation Conference (GECCO'99)*, pp. 657–664.

Tu, Z., and Lu, Y. (2004). A robust stochastic genetic algorithm (StGA) for global numerical optimization. *IEEE Transactions on Evolutionary Computation*, 8(5):456–470.

Wang, Y., and Dang, C. (2007). An evolutionary algorithm for global optimization based on level-set evolution and latin squares. *IEEE Transactions on Evolutionary Computation*, 11(5):579–595.

Yao, X., and Liu, Y. (1999). Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation*, 3(2):82–102.

Zhang, Q., Sun, J., and Tsang, E. (2005). Evolutionary algorithm with the guided mutation for the maximum clique problem. *IEEE Transactions on Evolutionary Computation*, 9(2):192–200.

Zhang, Q., Sun, J., Tsang, E., and Ford, J. (2003). Hybrid estimation of distribution algorithm for global optimisation. *Engineering Computations*, 21(1):91–107.

Zhang, Q., Sun, J., Tsang, E., and Ford, J. (2004). Combination of guided local search and estimation of distribution algorithm for solving quadratic assignment problem. In *Proceedings of the Bird of a Feather Workshops, Genetic and Evolutionary Computation Conference*, pp. 42–48.

Zhong, W., Liu, J., Xue, M., and Jiao, L. (2004). A multiagent genetic algorithm for global numerical optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 34(2):1128–1141.

Appendix A

Table A1 lists the benchmark functions used in our study. Functions f_1 through f_{25} are the same functions as F_1 through F_{25} defined in Suganthan et al. (2005), respectively.

Table A1: The basic characteristics of the test functions used in Noman and Iba (2008) and Suganthan et al. (2005). f is the global optimum. In the table, properties such as epistasis (Epi), multi-modal (Mul), and discontinuous (Disc), are also shown.

			Characteristics			
$f(x)$	f^*	n	Epi	Mul	Disc	Function description
f_1	–450	≤ 50	None	None	None	Shifted sphere function
f_2	–450	≤ 50	High	None	None	Shifted Schwefel's problem 1.2
f_3	–450	≤ 50	None	None	None	Shifted rotated high conditioned elliptic function
f_4	–450	≤ 50	High	High	None	f_{12} with noise in fitness
f_5	–310	≤ 50	None	High	Medium	Schwefel's problem 2.6 with global optimum on bounds
f_6	390	≤ 50	High	High	None	Shifted Rosenbrock's function
f_7	–180	≤ 50	Weak	High	None	Shifted rotated f_4 without bounds
f_8	–140	≤ 50	High	Weak	None	Shifted rotated Ackley's function with global optimum on bounds
f_9	–330	≤ 50	None	High	None	Shifted Rastrigin's function
f_{10}	–330	≤ 50	High	High	None	Shifted rotated Rastrigin's function
f_{11}	90	≤ 50	High	High	None	Shifted rotated Weierstrass function
f_{12}	–460	≤ 50	Weak	Weak	None	Schwefel's problem 2.13
f_{13}	–130	≤ 50	High	High	None	Shifted expanded Griewank's plus Rosenbrock's function
f_{14}	–300	≤ 50	High	High	None	Shifted rotated expanded f_{16} function
f_{15}	120	≤ 50	High	High	Medium	Hybrid composition function
f_{16}	120	≤ 50	High	High	Medium	Rotated version of f_{25}
f_{17}	120	≤ 50	High	High	Medium	f_{26} with noise in fitness
f_{18}	10	≤ 50	High	High	Medium	Rotated hybrid composition function
f_{19}	10	≤ 50	High	High	Medium	Rotated hybrid composition function with narrow basin global optimum
f_{20}	10	≤ 50	High	High	Medium	Rotated hybrid composition function with global optimum on the bounds
f_{21}	360	≤ 50	High	High	Medium	Rotated hybrid composition function
f_{22}	360	≤ 50	High	High	Medium	Rotated hybrid composition function with high condition number matrix
f_{23}	360	≤ 50	High	High	Medium	Noncontinuous rotated hybrid composition function
f_{24}	260	≤ 50	High	High	Medium	Rotated hybrid composition function
f_{25}	260	≤ 50	High	High	Medium	Rotated hybrid composition function without bounds
f_{26}	0	30	High	High	None	Salomon's function
f_{27}	0	30	High	High	None	Whitely's function
f_{28}	0	30	None	None	None	Sphere function
f_{29}	0	30	None	High	None	Ackley's function
f_{30}	0	30	None	High	None	Griewank's function
f_{31}	0	30	None	High	None	Rastrigin's function
f_{32}	0	30	None	High	None	Generalized Schwefel's problem 2.26
f_{33}	0	30	None	High	None	Generalized penalized function 1
f_{34}	0	30	None	High	None	Generalized penalized function 2
f_{35}	0	30	None	High	None	Schwefel's problem 2.22
f_{36}	0	30	None	High	None	Schwefel's problem 1.2
f_{37}	0	30/100	High	High	None	Rosenbrock's function
f_{38}	0	100	None	High	None	Schwefel's problem 2.21
f_{39}	–99.2784	100	High	High	None	f_7 in Leung and Wang (2011)
f_{40}	–78.33236	100	High	High	None	f_9 in Leung and Wang (2011)