Applied Soft Computing 40 (2016) 654–673

Contents lists available at ScienceDirect

Applied Soft Computing

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

Which algorithm should I choose: An evolutionary algorithm portfolio approach

Shiu Yin Yuen [a,*], Chi Kin Chow [a], Xin Zhang [b], Yang Lou [a]

[a] Department of Electronic Engineering, City University of Hong Kong, Hong Kong, China
[b] College of Electronic and Communication Engineering, Tianjin Normal University, Tianjin, China

ARTICLE INFO

Article history:
Received 18 July 2012
Received in revised form 6 May 2015
Accepted 14 December 2015
Available online 21 December 2015

Keywords:
Multi-method search
Algorithm portfolio
Performance prediction
Synergy
Scalability
Real world application

ABSTRACT

Many good evolutionary algorithms have been proposed in the past. However, frequently, the question arises that given a problem, one is at a loss of which algorithm to choose. In this paper, we propose a novel algorithm portfolio approach to address the above problem for single objective optimization. A portfolio of evolutionary algorithms is first formed. Covariance Matrix Adaptation Evolution Strategy (CMA-ES), History driven Evolutionary Algorithm (HdEA), Particle Swarm Optimization (PSO2011) and Self adaptive Differential Evolution (SaDE) are chosen as component algorithms. Each algorithm runs independently with no information exchange. At any point in time, the algorithm with the best predicted performance is run for one generation, after which the performance is predicted again. The best algorithm runs for the next generation, and the process goes on. In this way, algorithms switch automatically as a function of the computational budget. This novel algorithm is named Multiple Evolutionary Algorithm (MultiEA). The predictor we introduced has the nice property of being parameter-less, and algorithms switch automatically as a function of budget. The following contributions are made: (1) experimental results on 24 benchmark functions show that MultiEA outperforms (i) Multialgorithm Genetically Adaptive Method for Single Objective Optimization (AMALGAM-SO); (ii) Population-based Algorithm Portfolio (PAP); (iii) a multiple algorithm approach which chooses an algorithm randomly (RandEA); and (iv) a multiple algorithm approach which divides the computational budget evenly and execute all algorithms in parallel (ExhEA). This shows that it outperforms existing portfolio approaches and the predictor is functioning well. (2) Moreover, a neck to neck comparison of MultiEA with CMA-ES, HdEA, PSO2011, and SaDE is also made. Experimental results show that the performance of MultiEA is very competitive. In particular, MultiEA, being a portfolio algorithm, is sometimes even better than all its individual algorithms, and has more robust performance. (3) Furthermore, a positive synergic effect is discovered, namely, MultiEA can sometimes perform better than the sum of its individual EAs. This gives interesting insights into why an algorithm portfolio is a good approach. (4) It is found that MultiEA scales as well as the best algorithm in the portfolio. This suggests that MultiEA scales up nicely, which is a desirable algorithmic feature. (5) Finally, the performance of MultiEA is investigated on a real world problem. It is found that MultiEA can select the most suitable algorithm for the problem and is much better than choosing algorithms randomly.

1. Introduction

Rapid advances in Evolutionary Computation (EC) have been witnessed in the past two decades. There are now many powerful Evolutionary Algorithms (EAs) that are applied to scientific and engineering applications to find good quality solutions for challenging optimization problems. Famous examples include Genetic Algorithm (GA), Evolution Strategy (ES), Evolutionary Programming (EP), Particle Swarm Optimization (PSO), Differential Evolution (DE), Ant Colony Optimization (ACO), Artificial Immune System (AIS), Cultural Algorithm (CA), Estimation of Distribution Algorithm (EDA), Artificial Bee Colony algorithm (ABC), Biogeography Based Optimization (BBO), and others [1].

In spite of the proliferation of algorithms that use the evolution metaphor, for general users dealing with an optimization scenario, there is little readily available guideline of *which algorithm to choose*. Frequently, one resorts to words of mouth or fame of the algorithm, or try it out one by one in an exhaustive manner. The

* Corresponding author. Tel.: +852 34427717.
E-mail addresses: kelviny.ee@cityu.edu.hk (S.Y. Yuen), chowchi@hku.hk (C.K. Chow), xinzhang9-c@my.cityu.edu.hk (X. Zhang), felix.lou@my.cityu.edu.hk (Y. Lou).

http://dx.doi.org/10.1016/j.asoc.2015.12.021
1568-4946/

problem is compounded by the fact that individual algorithms will need parameter tuning to obtain the best performance, which is computational expensive or even prohibitive [2]. The current state of affairs motivates this paper.

Usually an algorithm has a standard recommended set of parameters defined by the researchers. This set of parameters is usually arrived at after many tests on benchmark functions and practical applications. Thus for each algorithm, we use the recommended set of parameters and do not attempt the challenging problem of parameter tuning and control [2]. Instead, we believe that the multiple algorithms can be complementary: an algorithm which does not work well on one problem will be replaced by an algorithm that works well for it. So the problem is how to select algorithms.

Note also that the question of choice of algorithm should be a function of the computational budget. For example, one algorithm may converge fast to a shallow local optimum, another may converge slower but to a deeper local optimum given enough time, still another may converge the slowest but eventually reach the global optimum. Which algorithm should one choose? If fitness evaluations are expensive, then only a small computational budget is allowed and the first one should be chosen. If fitness evaluations are relatively inexpensive or we have a design problem such that we can tolerate longer runs, then the second algorithm should be chosen. Finally, if fitness evaluations are cheap and we aim at solving a scientific problem of which finding the global optimum is essential, then the third algorithm should be chosen.

An algorithm portfolio approach is advocated to tackle the problem. Our conceptual framework is simple: (1) put promising EAs together in a portfolio; (2) an initialization is conducted in which each algorithm is run for some number of generations until there is a change in fitness; (3) use a predictive measure to predict the performance of each algorithm at the nearest common future point; (4) select the algorithm which has the best predicted performance to run for one generation; and (5) repeat step (3) and (4) until a given computational budget is reached.

Note that as the algorithm which has the best predicted performance may be different at different stages of the search, our approach will switch from one algorithm to another automatically and seamlessly.

In a nutshell, we propose to choose, at any point of the search, the algorithm which has the best predicted performance to run (for one generation). A typical scenario of running our algorithm is that after some trials in which each algorithm runs in parallel and interacts indirectly, an algorithm which has the best predicted performance by a considerable margin stands out, and only it is run for quite some time. If it is a very good algorithm that excels in small, medium and large budgets, then only it will run from then on. However, if it is an algorithm that converges fast to a local optimum, as discussed above, then it will run for awhile and gradually the predicted performance will not be as good compared with other algorithms. At which time a second algorithm, which has a better predicted performance, will take over. Like changes would occur as the search progresses.

A novel online performance prediction metric is proposed to advise which algorithm should be chosen to generate the next new generation of solutions. The metric is *parameter-less* – it does not introduce any new control parameter to the system – thus avoiding the difficult parameter tuning and control problem [2]. We name our algorithm Multiple Evolutionary Algorithm (MultiEA). It is designed for single objective optimization.

We choose four algorithms to compose the portfolio of MultiEA. They are (1) Covariance Matrix Adaptation Evolution Strategy (CMA-ES); (2) History driven Evolutionary Algorithm (HdEA); (3) Particle Swarm Optimization (PSO2011); and (4) Self adaptive Differential Evolution (SaDE).

These algorithms are chosen because they represent current state of the art methods:

CMA-ES [3] is one of the most powerful EAs available. It adapts its search strategy by evolving the covariance matrix of the current solutions. The idea is to increase the variance of the directions in which the search is successful and vice versa. A nice fundamental property which is unique to CMA-ES is its invariance to linear transformations of the search space.

HdEA [4] is a novel EA that uses the entire search history to make decision. The history is stored in a binary space partitioning (BSP) tree. When a new solution is generated by the EA, it finds the region it is contained within efficiently by traversing the tree to its leaf node. The local gradient is approximated by the local history information in the tree and a parameter-less gradient descent is executed to find a mutated solution. The performance of HdEA is tested on thirty four benchmark functions with dimensions ranging from 2 to 40. It outperforms eight benchmark evolutionary algorithms, which includes a Real Coded Genetic Algorithm (RCGA), classic DE, two improved DEs, CMA-ES, two improved PSOs, and EDA.

PSO introduces a new search paradigm which simulates the swarm behavior in birds and other animals. The velocity of each individual particle is modulated by both the historical local best position found by the particle and the historical global best position found by the whole swarm. Since its inception, two revised "standards" have been promulgated: PSO2007 and PSO2011. PSO2007 abandons the global best position concept. Instead, each individual chooses its own set of informants, and follows its local best as well as the global best amongst the informants. The idea is to distribute the search effort to avoid prematurely converging on the global best. It performs much better than the classic PSO. The PSO2011 [5] further improves PSO2007 by making the PSO more immune to linear transformations in the search space.

DE is also a powerful EA. Recently improved DE variants have won many EC competitions [6]. A fundamental idea in DE is to use the sum of a vector and the scaled difference of another two vectors to form the mutant vector. The recently proposed SaDE [7] employs four popular DE mutation strategies. During any one generation, DE maintains a set of trial vectors in its population pool. For each trial vector, a dice is rolled to select a DE strategy. Initially, all strategies have the same probability of being selected. Records are kept of the outcome of employing the strategies. A strategy is regarded as successful if it produces an offspring that survives into the next generation; otherwise, it is regarded as a failure. The success probability for each strategy is computed for the previous LP generations, where LP is the user defined learning period. The probability of selecting a particular strategy is proportional to its success probability. This probability is lower bounded by a small constant to preclude a strategy of attaining zero success probability and henceforth eliminated from all future considerations. SaDE shows remarkable performance. It outperforms classic DE and several recent adaptive DEs.

Though these are good EAs, it is also well acknowledged that they may not perform well for certain functions. For example, DE is known to be doing less well for non-separable functions [8] of which CMA-ES excels, and it is well known that CMA-ES performs poorly for the Rastrigin function if no artificial remedy such as enlarging the population is taken [9]. Finally, by the well known no free lunch (NFL) theorems, it is unlikely to find an algorithm which would do well for all problems [10,11]. So it would be of interest to see whether a portfolio approach can be successful in identifying the *correct* algorithm for a problem.

Experiments are conducted on 24 benchmark functions [12]. The effectiveness of MultiEA is demonstrated by comparing with four multiple algorithm approaches. They are (i) Multialgorithm Genetically Adaptive Method for Single-Objective Optimization

(AMALGAM-SO) [13]; (ii) Population-based Algorithm Portfolio (PAP) [14]; (iii) a multiple algorithm approach which randomly chooses an EA in each run (RandEA); and iv) a multiple algorithm approach with multiple algorithms running in parallel, in which the total budget is shared evenly amongst the algorithms (ExhEA). It is found that MultiEA outperforms the four algorithms.

Experiments are also conducted to compare MultiEA with the four individual algorithms (i.e., CMA-ES, HdEA, PSO2011, SaDE) in the portfolio in a neck to neck manner. This is a challenging test for MultiEA for the following reason: Given a problem, there will be an algorithm amongst the four which performs the best for *that* problem [10], though we do *not* know which. For MultiEA, it inevitably has to "waste" some fitness evaluations before deciding on which is the best and then execute it. Thus our aim is to test whether MultiEA is a competitive algorithm if viewed as a standalone EA. Of course, if it is, we have good reason to adopt MultiEA as it can literally combine the strengths and avoids the pitfalls of its component EAs. It is found that MultiEA is very competitive in the neck to neck comparison.

Through this experiment, an interesting positive synergic effect is discovered, namely, MultiEA can sometimes perform better than the sum of its individual EAs. This synergic effect embodies the principle of "one plus one is larger than two" or "the whole is larger than the sum of parts".

The scalability of MultiEA is studied. Since MultiEA is composed of individual algorithms, the best we should hope for is that it scales as well as the best algorithm in the portfolio. We found that this is indeed the case.

Finally, MultiEA is applied to a real world problem. It is found that MultiEA is able to select the most suitable algorithm for the problem.

The rest of the paper is organized as follows: Section 2 reviews past works on this problem. Section 3 introduces the MuliEA and the new performance prediction metric. Section 4 reports the experimental results. Section 5 gives the conclusions, limitations of the work, and directions for future research.

2. Past works

2.1. Algorithm portfolio

The idea that combining multiple algorithms into a portfolio may result in superior performance than using a single algorithm and that it is possible to improve performance by learning dates back to 1963 [15].

Huberman et al. [16] studied the benefits of using a portfolio of algorithms from an economical viewpoint by making analogies with a financial investment portfolio with a user defined overall risk. Here, stochastic algorithms that always find the correct solution are considered (i.e., Las Vegas algorithms). The risk of an algorithm is intuitively defined as its variance. Using the same computational time budget, an algorithm portfolio that allocates fractions of the budget to each algorithm is compared with a single algorithm that uses the whole budget. Assuming that the probability distribution of the time to obtain the correct solution is known for each algorithm, they report a systematic method of allocating the fractions that can obtain simultaneously lower expected time and lower risk. Other early multiple agent architectures include the A-Teams [17], which share a common population.

2.2. Multiple evolutionary algorithm approaches

Memetic algorithm (MA) [18] uses a hybrid approach. An EA is run in the background. Periodically, another algorithm is applied to improve selected individuals. This algorithm is usually a problem specific local improvement heuristic. The class of MAs can be considered as a two algorithm approach, with algorithms applied in sequence.

An interesting related paradigm is algorithm selection [19]. It aims to extract some distinguishing features from the problem. Machine learning techniques are used to map these features to the most suitable algorithm (e.g. see [20]).

Fukunaga [21] considered an algorithm portfolio with independent algorithms, i.e., there is no information exchange between algorithms. The utility of an algorithm is defined as the expected value of the best fitness found, subtracted by the weighted variance, with a user defined weighting factor. He considered optimizing the portfolio for tackling problems in a user specified problem class. The total computation time is divided into small, equally sized units. Several instances of the problem are run, and the utility of each algorithm as a function of units allocated is recorded in a performance database. The recorded information is used to build up a bootstrap probability distribution. Using the distribution, all possible combinations of the algorithm portfolio are then evaluated and the optimal algorithm portfolio is found for the problem class. The multiple algorithms are GAs with different control parameter values. The approach is applied to the traveling salesman problem. It is found that the portfolio approach performs better than a single GA optimized using parameter tuning.

Gagliolo et al. [22] reported an interesting approach. Their portfolio consists of a set of GA with different parameters. The convergence curve of each GA is plotted, and the curve is extrapolated using linear regression with an adaptive sliding window to estimate the time needed to reach a user-defined fitness level. A unit of resource is distributed by an allocator such that GA with small estimated time and has been run for substantial time is given more resources. All GAs are then run in parallel using the resource allocated. Then the process iterates.

Vrugt et al. [13] reported a self adaptive multi-method search known as AMALGAM-SO. The idea is to use a set of q EAs $\{A_1, \ldots, A_q\}$ that have a common population with N individuals. Initially, the number of offspring $\{N_1, \ldots, N_q\}$ that each algorithm produces is fixed by the user, where $\sum_{i=1}^{q} N_i = N$. After the N offspring are generated, the N parents and the N offspring are sorted by decreasing fitness. Individuals with higher fitness are inserted into the population pool of the next generation one by one. To maintain diversity, an individual will be inserted only if its Euclidean distance is larger than a user-specified threshold compared with all existing individuals in the pool. The process goes on until N new individuals are generated, which makes up the next generation. Each algorithm has its own stopping criterion. The multimethod search stops when one of the stopping criteria is fulfilled. Then the search is re-run with the population size doubled and the number of offspring is recalculated as $N_i = N\Psi_i / \sum_{i=1}^{q} \psi_k$, where Ψ_i is the number of generations for which algorithm i is responsible for fitness improvement. To prevent an algorithm to be driven extinct, the minimum N_i/N ratio is fixed by the user. To pass information from one run to the other, the best found solution is included as one of the individuals in the otherwise randomly generated initial population of the next run, provided that a numerical condition is fulfilled. On simple uni-modal problems, the method obtains similar performance as existing individual algorithms. For complex multi-modal problems, the method is superior.

Peng et al. [14] proposed an algorithm called Population-based Algorithm Portfolio (PAP). q EAs are run in parallel, each with their own populations. There are two important parameters in the algorithm: migration interval I and migration size s. A migration across population occurs every I generations. The migration process is as follows: for each algorithm, the s best individuals are found amongst all the individuals in the remaining $(q-1)$ populations. These s individuals are appended to the population. Then the worst

s individuals in the population are discarded. It is experimentally found that PAP performs better than running each individual algorithm with the same computational budget. The authors conclude that the performance gain is due to the synergy of the algorithms through the migration process.

For multiobjective problems, Vrugt and Robinson [23] proposed a self adaptive multi-method search named AMALGAM. A common population and a set of algorithms are used. In each generation, for each algorithm i, the number of offspring N_i that successfully goes into the next generation is recorded. The number of offspring that each algorithm can generate in the next generation is made proportional to N_i, that is, proportional to its reproductive success rate. To avoid an algorithm becoming extinct, an algorithm is guaranteed to generate a user-defined minimum number of offspring. In common with [23], Grobler et al. [24] also used a common population. They studied single objective problems. In their method, each individual in the population may select different algorithms to produce offspring. Different selection methods are compared in [24].

Recently, Yuen and Zhang [25,26] and independently Tang et al. [27] propose an original research problem: given a known set of problems, how to find an automatic method to compose a portfolio. Clearly, the composition result is a function of the portfolio algorithm used. In [25,26], the algorithm proposed in this paper is used and in [27], the PAP is used.

2.3. Algorithm selection and credit assignment

Within the context of single algorithm, parameter control techniques [2] have been extensively investigated. Under this branch of research, operator selection and its associated credit assignment have been investigated. In the context of multiple algorithms with interaction, operator selection has some analogies with algorithm selection. The researches in the above have mainly used probability matching for algorithm selection. Let $p_{\min}$ be the user defined minimum probability for an operator. A known problem of probability matching is that with q operators, the maximum probability of selecting an operator $p_{\max}$ is $1-(q-1)p_{\min}$. It degrades the performance because the best operator that delivers the highest expected reward should *always* be chosen, instead of being chosen with a probability [28]. Thierens [28] addressed this problem by using a winner take all strategy; the best algorithm is moved toward $p_{\max}$, while others are uniformly moved toward $p_{\min}$. The movement rate is governed by a user defined greediness factor. Calculations and experiments in [28] find that for a large range of values of this factor, it is better than using probability matching. Recently, Fialho et al. [29] extended this idea by formulating the problem as a dynamic multi-armed bandit problem [30,31]. At any one instance, the algorithm to execute is the one which has the highest average reward so far. Unfortunately, the model is quite complicated and as the reward distribution may change over time, an additional change detection method is needed. Finally, it is unclear whether the multi-armed bandit paradigm is an appropriate model for algorithm selection.

Recently, an online racing algorithm, Max-Race Portfolio (MRP), is proposed to select algorithms in a portfolio [32]. Different from other existing approaches, the racing algorithm aims to pinpoint algorithms that are statistically unlikely to succeed online. When sufficient statistical evidence is obtained, that algorithm is removed from the portfolio. The statistical model is based on extreme value theory, which in itself has several user-defined parameters which require tuning. Another potential problem on all racing approach is that it permanently removes an algorithm, which may later be found to be very useful when there is a larger budget. Experimental results in [32] show that MRP outperforms AMALGAM-SO, PAP and MultiEA on problems generated by a Gaussian landscape generator (using a different suite of algorithms to compose the portfolio and different experimental settings.) The performance remains to be verified for problems which do not share a common way of generation. In particular, whether the parameters in the extreme value theory are problem dependent needs to be further investigated.

2.4. Limitations of existing approaches

1. Early algorithm portfolio approaches (e.g. [16]) are designed for Las Vegas algorithms. However, the usual applications of EA do not require or guarantee the finding of the global optimal solution (i.e., they are Monte Carlo algorithms).
2. Some algorithm portfolio approaches (e.g. [21]) require a problem class to have been well defined and previous history is used to advise the algorithm selection. This is not applicable to the scenario in which a multiple algorithm EA is requested to handle a *single problem instance* with little or no applicable prior knowledge.
3. All existing algorithms invariably introduce more *control parameters* into the algorithm (e.g. the risk in [16], the initial number of offspring for each algorithm, the minimum probability, diversity threshold, stopping criteria for each algorithm, and the condition for the best found solution in [13], the two migration parameters in [14], the resource to be allocated in each iteration in [22], the minimum number of offspring [23], the number of iterations elapsed before the next selection [24], the parameters in the extreme value theory [32]). A careful tuning of these control parameters is needed. Such tuning is computationally expensive [2]. Even when this has been done, there is no guarantee that it would work well in a new unknown problem [11].
4. Many multiple algorithm researches have used probability matching for algorithm selection, whereas some recent research [28] argues that it is not the best approach. On the other hand, the related researches within parameter control introduce new methods for operator selection and credit assignment [2]. This involves introducing new control parameters, which itself requires a non-trivial parameter tuning.
5. While recent researches [13,14,23,24] have demonstrated that a multiple algorithm approach is promising, a *self adaptive* approach is taken, i.e., incorporate the algorithm selection decision within the evolutionary process. It is known that a self adaptive approach does not mitigate completely the problem of premature convergence, as the whole population may be focused on only a part of the search space which then biases the future available search strategies. Moreover, offspring generated from different algorithms may mislead each other.
6. Another disadvantage of the self adaptive approach is that it sheds no physical insight on *why* an algorithm is good for a problem, as it does not have an easily understood physical model for credit assignment. Also, the process sheds no new insight on *which* algorithm is good for *which* problem.

2.5. Merits of our approach

In this research, we present a novel alternative approach which overcomes the above limitations. It does not require the global optimal solution to be known; it works on a single problem instance; it does not introduce any new control parameter; it uses a winner take all strategy rather than probability matching; as the algorithms run independently, it does not suffer from the limitations of a self adaptive approach, and offspring from different algorithms will not mislead each other. An easily understood physical model for credit assignment based on fitness prediction is introduced. It also gives direct insights on "which algorithm is good for which problem" as a function of computational budget.

Existing multiple algorithm approaches [13,14,23,24] find a good *synergy* by self adaptive information exchange between

algorithms through a common population. The present approach aims to *select* the best algorithm at *each* instance by predicted future performance, based on past history. It switches between algorithms dynamically as new search information is obtained which enables revised predictions to be made. A synergy is achieved indirectly. Our approach is complementary to existing works.

The most closely related approach to the present paper is [22]. Both use predicted performance to allocate resources, and both advocate using algorithms which are independent and do not interact with each other. However, in our approach, we do not require the user defined required fitness level, which in general is impossible to know a priori. Moreover, we do not extrapolate to a far future; we merely predict at the nearest common future point. The rationale is that we reckon that prediction into the far future is very difficult and error prone (see Fig. 2). Furthermore, there is also a difference in design philosophy. Our algorithm does not allocate a fixed amount of resource and distribute to the component algorithms, which are run in parallel; we allocate all resources to the algorithm which is predicted to perform best at the common future point, execute that algorithm one generation and then iterates and predicts again. This allows prediction to be updated as soon as new data arrives. Finally, instead of an adaptive sliding window approach, we attempt to use all historical data in the convergence curve of the algorithm to make a better prediction.

Recently, we have reported the preliminary result of this paper in [33]. Instead of the benchmarking suite [12], we have selected another set of algorithms to form our portfolio and applied MultiEA to another widely used benchmarking suite CEC 2005 [34]. The conclusions reached corroborate well with the result in this paper. In particular, it is found that MultiEA is highly competitive in a neck to neck comparison with the individual algorithms which form the portfolio. Also, MultiEA is superior over a baseline EA which randomly selects algorithms to execute. It is also found that MultiEA outperforms AMALGAM-SO and PAP.

In this paper, we have applied MultiEA to the benchmarking suite [12] with a portfolio composed of a different set of algorithms. The experimental results agree with that in [33]. This gives an independent verification that the findings in this paper are transferrable to other problem-algorithm contexts.

Section 4.5.1 compares MultiEA with other existing algorithm portfolios, namely, AMALGAM-SO and PAP, as well as two baseline algorithm portfolios which randomly selects algorithms. Section 4.5.2 is a neck to neck comparison of MultiEA with individual algorithms that make up the portfolio.

The following new results are also reported in this paper: a positive synergic effect of MultiEA is reported in Section 4.5.3. The scalability of MultiEA is studied in Section 4.5.4. The performance of MultiEA when applied to a real world application is investigated in Section 4.5.5.

3. A new evolutionary algorithm portfolio approach

3.1. Predictive performance measure

This research focuses on single objective continuous parameter optimization with dimension D. Without loss of generality, let the optimization be a minimization, and assume one has a problem P for which we have no priori knowledge about which algorithm is the best. Assume that q EAs have been chosen and placed within a portfolio $AP = \{A_1, \ldots, A_q\}$. The q algorithms are executed independently and there is no information exchange between them.

Let α_i be the number of generations that algorithm A_i has run. Let $f_i(j)$ be the best (i.e., smallest) fitness found by A_i at generation j. Whether the best so far solution participates in the evolution, it is reasonable to keep a copy of it. Hence $f_i(k) \leq f_i(j)$ iff $k \geq j$. An

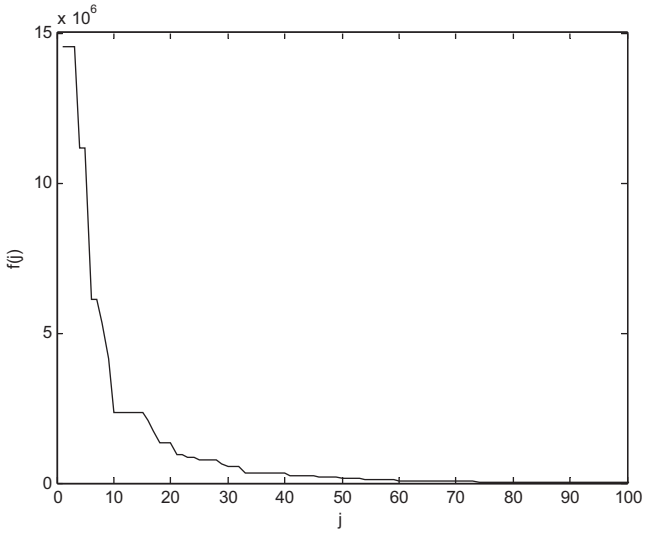

Fig. 1. A typical convergence curve: it is the convergence curve of SaDE on optimizing function f_2. The horizontal axis is the generation number and the vertical axis is the best found fitness value at each generation.

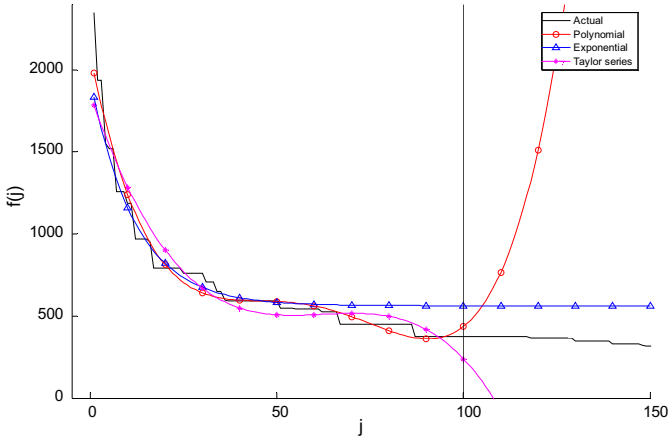

Fig. 2. Extrapolation using (a) exponential; (b) polynomial and (c) Taylor series. The horizontal axis is the generation number and the vertical axis is the best found fitness values at each generation. The actual data in the first 100 generations, delimited by the vertical dashed line, is used for the extrapolation. A custom built genetic algorithm is used in (a) and standard linear regression is used for (b) and (c).

indicator of the performance is the *convergence curve* $C_i = \{(j, f_i(j))\}$, which records the *entire history* of the convergence trend. A typical C_i is shown in Fig. 1.

The relative performance of the algorithms can be compared by comparing the convergence curves. This is also the standard practice in many EC papers. A popular measure used in the EC community is comparing $f_i(j)$ while keeping the total number of evaluations a constant. It simply means running each algorithm by a fixed user defined number of evaluations and then chooses the winner. This exhaustive approach is very computationally expensive. It can be considered as a generalized form of parameter tuning, only that in this case, the parameter is replaced by algorithm. Parameter tuning is useful if P is a *typical* problem of the class of problems to be solved. However, in practical optimization scenarios, usually the class of problems is not well defined, and it is impossible to know beforehand what a *typical* problem is.

It is more reasonable to predict fitness at some future point common to all algorithms and choose the winner. This requires extrapolating the convergence curve. Fig. 2 shows extrapolation using fitting of (a) exponential; (b) polynomial and (c) Taylor series.

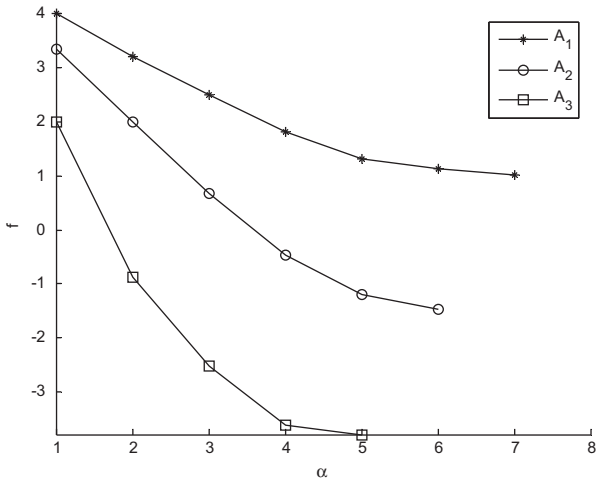

Fig. 3. The convergence curves of A_1, A_2 and A_3 in this example.

It shows some interesting characteristics of extrapolation using standard functions. If one wishes to extrapolate a little, then these functions perform well. However, if the extrapolation is far into the future, then these functions give bad results. Moreover, these extrapolations do not use the knowledge that the curve is non-increasing.

This paper proposes a novel prediction measure to tackle this problem. Let revisit the convergence curve $C = \{(j, f(j))\}$. For clarity, we have dropped the subscript i. Define $C(l) = \{(\alpha\text{-}l, f(\alpha\text{-}l)), \ldots, (\alpha, f(\alpha))\}$as the sub-curve which includes all the data points from the $(\alpha\text{-}l)$th generation to the αth generation, for which l is the *history length*. For each sub-curve $C(l)$, a linear regression is done to find

$$\underset{(l_1,l_2)}{arg\ min} \sum_{(x,y)\in C(l)} (y - (ax + b))^2 \tag{1}$$

which gives a straight line with slope a and y-intercept b that minimizes the mean squared error. The predicted fitness using data points in $C(l)$ for future generation t is

$$pf(t, l) = at + b \tag{2}$$

Since l may take on values from 1 to $\alpha - 1$, we have $\alpha - 1$ predicted values $pf(t, 1)$ to $pf(t, \alpha - 1)$. We treat these predicted values as sample points of an unknown distribution, and fit a bootstrap probability distribution $bpt(t)$ to it. In this paper, the distribution is computed by Matlab statistical toolbox function *ksdensity*. (Strictly speaking, *ksdensity* has some parameters that can be varied; in this paper, we have used the standard values provided in MATLAB for a "standard" interpolation.) We predict the fitness at t by sampling from $bpd(t)$ to get the predicted fitness $pf(t)$.

The underlying idea is to treat each linear regression model $C(l)$ *with equal probability*. The distribution model is designed to model the uncertainty in the prediction. In the special case that the convergence curve is indeed linear, then the bootstrap distribution will be a single spike. Since more recent points appear in more regression models, implicitly a heavier weighting is put on more recent points and vice versa.

Now for each algorithm i, the predicted best fitness $pf_i(t)\,(t > \alpha_i)$ *predicts* the fitness of the algorithm *if* it *were* evaluated t generations.

Assume that the population size m of the algorithms is equal. Then the nearest future t_i is $\alpha_i + 1$. This is the nearest future in which algorithm i has run one more generation. In this case, the unit of time is the number of generations. Let $t_{\min} = \max(t_1, \ldots, t_q)$. This is the nearest common future of all the algorithms. The algorithm which will generate the next generation is $\underset{i}{arg}\min_i(pf_i(t_{\min}))$, where $pf_i(t_{\min})$ for each i is sampled from its bootstrap probability distribution $bdp_i(t_{\min})$.

It has a clear physical meaning: The algorithm that will generate the next generation is one for which the predicted best fitness is the smallest if they were evaluated the *same* number of generations at the nearest common future point.

In general, let m_i be the population size of algorithm i. Then the nearest future is $m_i t_i$, of which $t_i = \alpha_i + 1$. This is the nearest future in which algorithm i has run one more generation. In this case the unit of time is the number of fitness evaluations. Let $t_{\min} \equiv (mt)_{\max} = \max(m_1 t_1, \ldots, m_q t_q)$. This is the nearest common future of all the algorithms. The algorithm which will generate the next generation is $\underset{i}{arg}\min_i(pf_i(t_{\min}))$, where $pf_i(t_{\min})$ for each i is sampled from its bootstrap probability distribution $bdp_i(t_{\min})$.

Note that only efficient algorithms will be selected often; inefficient algorithms will be selected sparsely since its predicted fitness is large. Also, note that the above metric is *parameter-less*. It does not introduce any additional parameter into the algorithm portfolio. Moreover, the metric uses a winner take all strategy, thus circumventing the problem of probability matching.

In the above, a linear regression is used for the estimation. A nonlinear kernel (e.g. an exponential function) may conceivably be used. However, finding the parameters of the nonlinear kernel in general requires a nonlinear optimization, which requires its own optimization methods. Such a method would also be much more computationally intensive. Thus we have decided to use the simpler linear kernel.

In principle, one can abstract the bootstrap probability distribution by its statistics (e.g. mean or median), and make the prediction deterministic. We do not take such an approach because such summary statistics throws away much useful information. For example, $\alpha_i - 1$ data points will be replaced by a single number (mean or median), which would throw away information contained in $\alpha_i - 2$ points. Also, a probability distribution is ideal to represent the fact that the prediction is by nature stochastic and uncertain.

In [33], we experiment with mean and median as alternative measures and confirm the above prediction that they are not good predictive measure. Amongst several heuristic measures that we have compared in [33], we find that the predictive measure reported in this paper has the best performance. Also, a predictive measure that uses the raw histogram instead of *ksdensity* to construct the probability distribution has only slightly worse performance. Note that the use of the raw histogram makes the algorithm truly parameter-less.

3.2. Initialization of individual algorithms

One has to ensure that each algorithm is able to make a non-trivial prediction. Thus each algorithm A_i is run until there is a decrease in the best fitness, i.e., until the smallest α_i such that $f_i(\alpha_i - 1) \neq f_i(\alpha_i)$. Note that the initial number of generations α_i is determined automatically, without introducing any extra control parameter.

3.3. An illustrative example

We illustrate the above concepts by a hypothetical scenario. In this example, MultiEA consists of three candidate EAs: A_1, A_2, and A_3, with population sizes $m_1 = 15$, $m_2 = 20$ and $m_3 = 30$, respectively. Suppose MultiEA has already run 18 generations, of which $\alpha_1 = 7$ generations is by A_1, $\alpha_3 = 6$ generations is by A_2 and $\alpha_3 = 5$ generations is by A_3. Thus A_1, A_2 and A_3 have been evaluated $m_1\alpha_1 = 105$, $m_2\alpha_2 = 120$ and $m_3\alpha_3 = 150$ solutions, respectively. Fig. 3 shows the convergence curves of the EAs. At this point, we have to make a choice of which algorithm to run next. As

$t_{min} = (mt)_{max} = \max(105+15, 120+20, 150+30) = 180$, we need to predict the fitness values of A_1, A_2 and A_3 after the 12th (180 evaluations), the 9th (180 evaluations) and the 6th generations (180 evaluations), respectively. In this case, the number of evaluations is the same for the three algorithms. If not, an algorithm is predicted at the number of generations which gives the number of evaluations smaller than but closest to 180.

The selection of EA for the 19th generation of MultiEA starts from obtaining the distributions of the predicted fitness values of A_1. Since A_1 has been executed 7 generations, we are going to formulate 6 prediction models $C_1(1), \ldots, C_1(6)$ for which 6 predicted fitness values $pf_1(t_{min}, a)$ ($t_{min} = 180$ evaluations, $a = 1, \ldots, 6$) are guessed (Fig. 4(a)–(f)). For each of the sub-figures, a (gray ○ or black ○) circle represents the best fitness value at each generation; a black circle represents the data points used for formulating the linear regression model at a particular history length l. The dashed line represents the linear regression prediction model. Symbol '☼' at each sub-figure represents the predicted fitness $pf_1(t_{min}, l)$. Using these 6 predicted fitness, we construct the bootstrap probability distribution $bdp_1(t_{min})$ (Fig. 5(a)).

We then repeat the previous procedures to construct the bootstrap distributions $bdp_2(t_{min})$ and $bdp_3(t_{min})$ of the estimated best fitness value of A_2 and A_3 at the 9th and the 6th generations (Fig. 5(b) and (c)). Afterwards, for each algorithm, the predicted fitness is randomly sampled from the corresponding probability density distribution. For example, suppose the random sampled values for A_1, A_2 and A_3 are $pf_1(t_{min}) = -1.5$, $pf_2(t_{min}) = -5.1$, and $pf_3(t_{min}) = -5.8$, respectively. Since $pf_3(t_{min})$ is the smallest, A_3 is executed a generation and its convergence curve is lengthened to $\alpha_3 = 6$. In the next generation of MultiEA, as the convergence curve of A_3 is lengthened, we have to re-formulate the prediction model of A_3, and construct the corresponding revised bootstrap distribution.

3.4. MultiEA algorithm in pseudo code

We summarize the proposed method by presenting it in algorithmic format after Fig. 5.

Algorithm (MultiEA)
Input: *A portfolio of q EAs $AP = \{A_1, \ldots, A_q\}$; A_i has population size m_i; single objective minimization problem P with dimension D; maximum number of evaluations N*
1. *for $i = 1$ to q*
   run A_i until there is a change of fitness. Let α_i be the number of generations that A_i has run
2. compute the nearest common future point $t_{min} \equiv (mt)_{max} = \max(m_1 t_1, \ldots m_q t_q)$, where $t_i = \alpha_i + 1$
3. *for $i = 1$ to q*
   {
   construct convergence curve $C_i = \{(j, f_i(j)) \,|\, j = 1, \ldots \alpha_i\}$
   construct sub-curves $\{C_i(1), \ldots, C_i(\alpha_i - 1)\}$
   for each sub-curve $C_i(l)$
   {
   least square line fit to get line parameters (a,b)
   predict the fitness at the smallest common future point $pf_i(t_{min}, l)$
   }
   use the $\alpha_i - 1$ sample points $pf_i(t_{min}, l)$ $(l = 1, \ldots, \alpha_i - 1)$ to construct bootstrap probability distribution $bpd_i(t_{min})$
   Sample $bpd_i(t_{min})$ to get $pf_i(t_{min})$
   }
4. choose the algorithm with index $\underset{i}{\arg\min}(pf_i(t_{min}))$, which has the best predicted performance
5. *run it for one generation. $\alpha_i \leftarrow \alpha_i + 1$*
6. *Record the best-so-far solution s_{best} found by the portfolio*
7. *Stop if total number of evaluations $\geq$ N. Otherwise goto Step 2*

Output
Best solution found s_{best}

4. Experimental results

4.1. Test problem set

MultiEA is examined using 24 real-parameter test functions: f_1–f_{24} [12]:

f_1	Sphere Function
f_2	Ellipsoidal Function
f_3	Rastrigin Function
f_4	Büche-Rastrigin Function
f_5	Linear Slope
f_6	Attractive Sector Function
f_7	Step Ellipsoidal Function
f_8	Rosenbrock Function, original
f_9	Rosenbrock Function, rotated
f_{10}	Ellipsoidal Function
f_{11}	Discus Function
f_{12}	Bent Cigar Function
f_{13}	Sharp Ridge Function
f_{14}	Different Powers Function
f_{15}	Rastrigin Function
f_{16}	Weierstrass Function
f_{17}	Schaffers f_7 Function
f_{18}	Schaffers f_7 Function, moderately ill-conditioned
f_{19}	Composite Griewank-Rosenbrock Function $f_8 f_2$
f_{20}	Schwefel Function
f_{21}	Gallagher's Gaussian 101-me Peaks Function
f_{22}	Gallagher's Gaussian 21-hi Peaks Function
f_{23}	Katsuura Function
f_{24}	Lunacek bi-Rastrigin Function

These 24 test functions are clustered into 5 subgroups: (1) f_1–f_5 are separable; (2) f_6–f_9 are with low or moderate conditioning; (3) f_{10}–f_{14} are unimodal with high conditioning; (4) f_{15}–f_{19} are with adequate global structure and (5) f_{20}–f_{24} are with weak global structure. Please refer to the description in [12] for full details.

The dimension D of all test functions is chosen to be 40. The search spaces of all functions are $[-5, 5]^{40}$. $[-5, 5]$ is the interval used in [12] and 40 is the largest dimension tested in [12].

Since the test algorithms are stochastic, the performance of an algorithm is measured by statistics gathered out of 100 independent runs.

Though [12] advocates changing the parameters of the functions in each run, we have elected not to follow exactly the methodology in [12] as we prefer to have a *fixed* function that illustrates our ideas better. It should be noted that these 24 functions are selected with the sole intention to illustrate general behavior of our proposed algorithm. One can view them as a selection of 24 *fixed* application problems with user defined absolute maximum computational

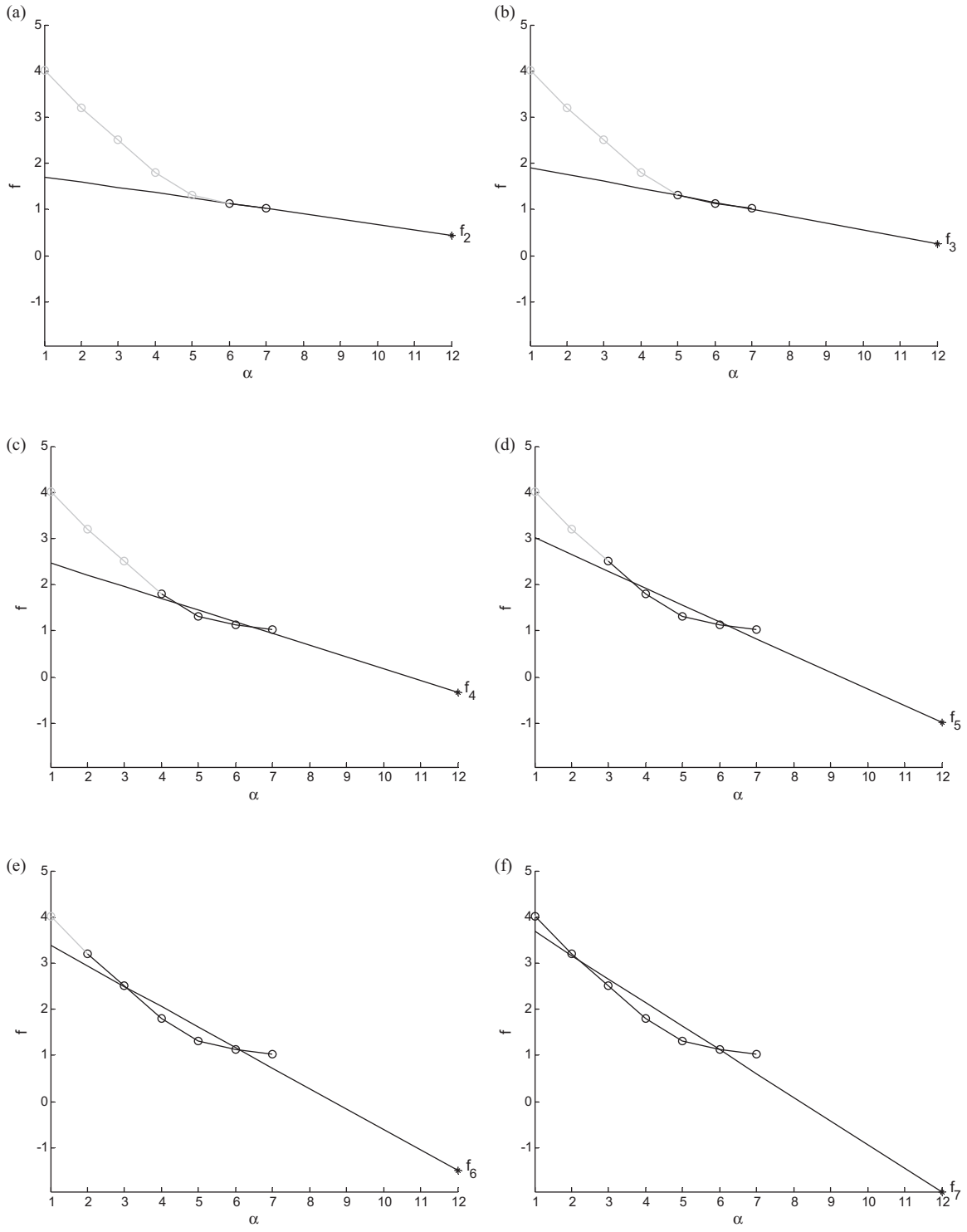

Fig. 4. An illustration of the estimation of the predicted best fitness. (a)–(f) The prediction by the latest 2–7 best fitness values.

budget (see Section 4.3 for details). Thus we have made the rotation matrix of each test problem the same in the 100 runs. In other words, the 100 runs are run on the same test problem.

4.2. Test algorithm set

To evaluate the performance of the proposed algorithm, we compare the performance of MultiEA with four multi-algorithms. The component algorithms in the portfolio for the four multi-algorithms are the same, i.e., $q = 4$ (A_1 = CMA-ES, A_2 = HdEA, A_3 = PSO2011, and A_4 = SaDE).

Test algorithm 1 – Multiple Evolutionary Algorithm (MultiEA).

Test algorithm 2 – Population-based Algorithm Portfolio (PAP) [14]: The migration size and the migration interval, as suggested in [14], are chosen to be 1 and $N/20$, respectively, where N is the number of fitness evaluations.

Test algorithm 3 – A Multialgorithm Genetically Adaptive Method for Single-Objective Optimization (AMALGAM-SO) [13]: The population size is chosen to be 100, following [13]. The

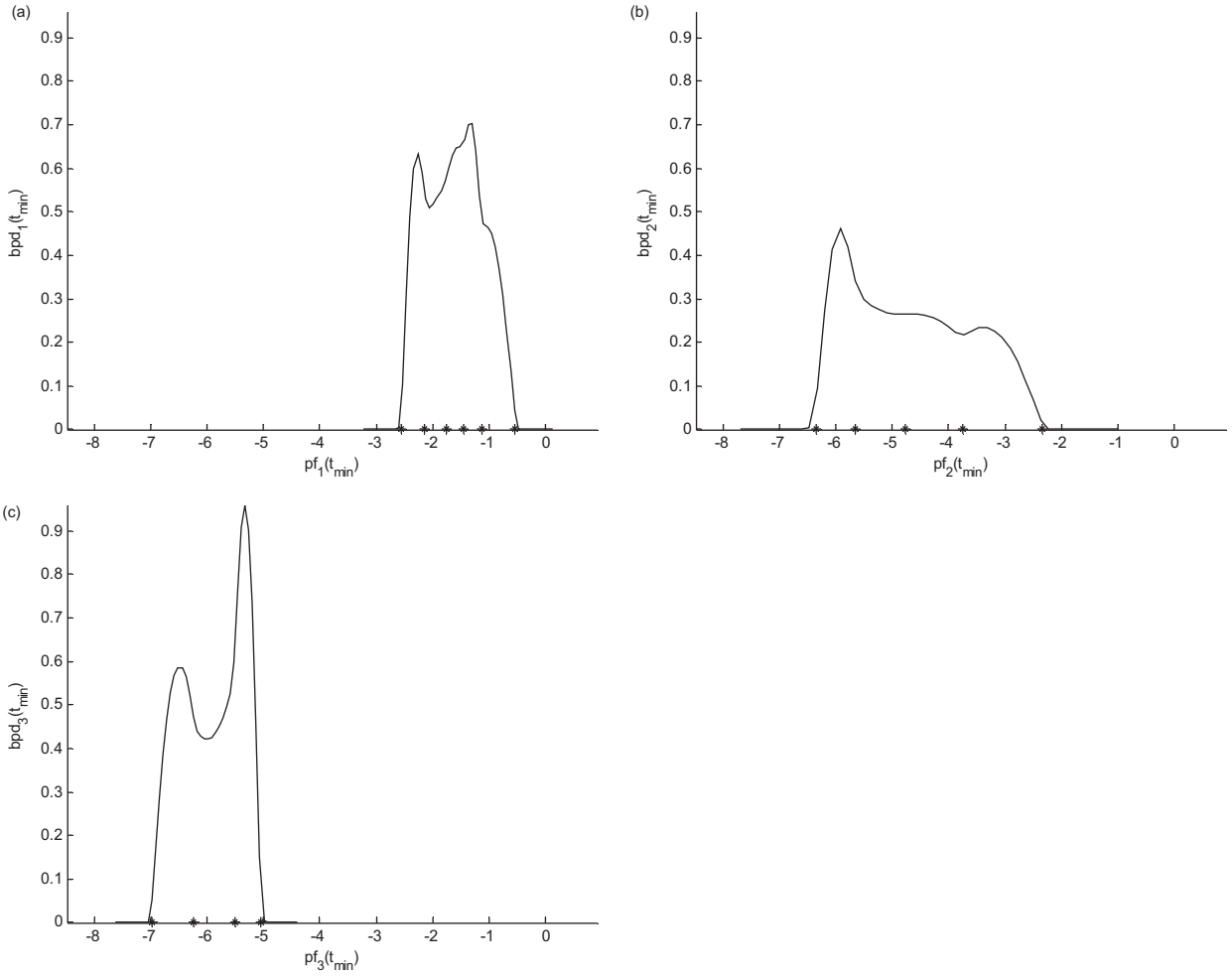

Fig. 5. The bootstrap distribution of the predicted best fitness value of (a) A_1, (b) A_2 and (c) A_3.

individuals are initially distributed to the EAs by the same strategy suggested in [13]. The restart feature of AMALGAM-SO and the boundary handling of the corresponding EAs follows the same rules as in [13]. Note that though source code is provided in [13], it cannot be used verbatim as the individual algorithms that compose the portfolio are different. Thus we have chosen to re-implement the algorithm, following faithfully the description of the algorithm in [13].

Test algorithm 4 – Randomly chosen Evolutionary Algorithm (RandEA): Given the number of evaluations N, if we have no knowledge about which algorithm amongst the q algorithms available is the best, it is reasonable to randomly select an algorithm and run it for N evaluations. This algorithm is referred to as RandEA. Its performance is defined as the *average* performance of the q algorithms.

In our implementation, in each run, we run all 4 EAs for N evaluations. Then calculate the averaged best fitness amongst the four to get the performance for one run. The performance is the averaged performance in 100 runs.

Test algorithm 5 – Exhaustive Evolutionary Algorithm (ExhEA): It is also reasonable to allocate N/q evaluations to each of the algorithms and run them in parallel. This algorithm is referred to as ExhEA. Its performance is determined by choosing the best fitness attained amongst these algorithms.

RandEA and ExhEA are multi algorithms, albeit very simple ones. They serve as baselines and also to test whether there is a positive synergy in MultiEA.

4.3. A novel performance evaluation measure

In the literature, algorithms are frequently compared when they have run for a *fixed* number of evaluations. Researchers have recognized this deficiency. In fact, in recent competitions, the performance of algorithms is evaluated at several numbers of evaluations. This gives a better picture.

One can say that algorithm A is a better algorithm compared with algorithm B if A obtains the target fitness (i.e., either globally or nearly globally optimal fitness) but B cannot. On the other hand, if *both* algorithms can obtain the target fitness within a user defined absolute maximum number of fitness evaluations, the one using fewer fitness evaluations to achieve this goal should be viewed as being better.

Due to the above reason, we introduce a novel evaluation measure. Our procedure is as follows:

We assume that all EAs in our portfolio can run at most $N_{\max}$ fitness evaluations, where $N_{\max}$ is a suitable user defined value. For each of the test algorithm A_i, we execute it for $N_{\max}$ fitness evaluations and the corresponding convergence curve is $C_i = \{(j, f_i(j)) \,|\, j = 1, \ldots N_{\max}\}$. Afterwards, we find the best fitness amongst the q algorithms $\max_i \{f_i(N_{\max})\}$. This best fitness will be our target fitness. Now we backtrack to find the minimum number of evaluations N_t ($N_t \leq N_{max}$) for which the first algorithm finds the target fitness. The performance of any algorithm A_k is now

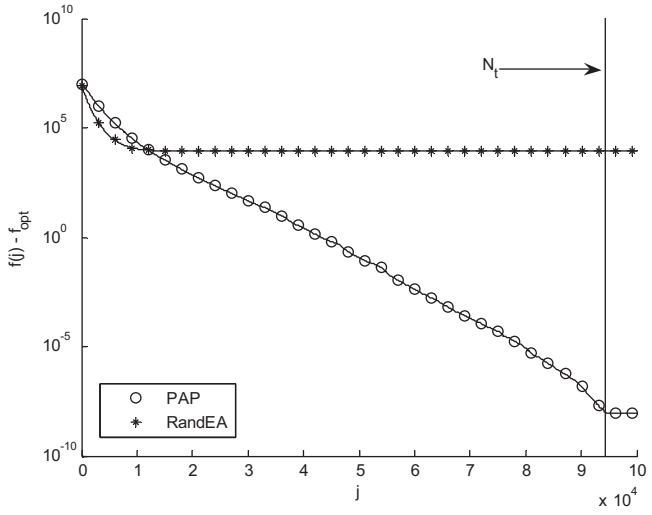

Fig. 6. Example of defining N_t.

Table 1
Settings of the four evolutionary algorithms in the portfolio.

CMA-ES	• Population size $\lambda = 4 + \lfloor 3\ln D \rfloor$
HdEA	• Population size = 20 • Uniform crossover operator with crossover rate $\gamma = 0.5$
PSO2011	• Swarm size $S = 40$ • $K = 3$ • $W = 0.721$ • $C = 1.193$ • $P = 1 - (1 - S^{-1})^K$
SaDE	• Population size = 40 • LP = 50 • $K = 4$

represented by $C_k = \{(j, f_k(j)) \,|\, j = 1, \ldots N_t\}$. This ensures that the algorithms are compared *exactly when* the first algorithm has successfully found the best solution possible to be found if given at most $N_{\max}$ fitness evaluations.

The physical meaning of $N_{\max}$ is the absolute maximum number of fitness evaluations that the user would allocate for a particular application. It has a strong physical meaning. For many engineering applications, a fitness evaluation is expensive and/or time consuming. For example, it may take from 1 s to 1 day for one fitness evaluation. Thus usually there is an upper limit to the absolute maximum number of fitness evaluations allowed. For example, if a design project requires to find the best design within 1 month, and it takes 25 s for one fitness evaluation, then $N_{\max} \leq (30)(24)(3600)/(25) = 103{,}680$. Understandably, we would like to find the design sooner than $N_{\max}$ evaluations if we can. Thus one would be interested in an algorithm which takes the minimum time to find the eventual best design (i.e., target fitness) and evaluate algorithms at exactly the instance that this design is found.

Fig. 6 illustrates the procedure of defining N_t. The curve with symbol 'O' represents the convergence curve of PAP and the curve with symbol '*' represents the convergence curve of RandEA. The target fitness is the best fitness found by PAP. Then we backtrack the convergence curve of PAP to determine N_t. The dashed line in Fig. 6 indicates the value of N_t (i.e. cut at around the 91,000th evaluation).

In the simulation results below, $N_{\max}$ is chosen to be 100,000.

4.4. Settings of evolutionary algorithms in the portfolio

In the portfolio, $q = 4$ (A_1 = CMA-ES, A_2 = HdEA, A_3 = PSO2011, and A_4 = SaDE). Table 1 lists the parameter settings of these algorithms. The parameter values follow their recommended settings in the literature. It is tacitly assumed that these algorithms have been "tuned" for best performance by their original designers for the "average" unknown real problem, which is not an unreasonable assumption. Note that AMALGAM-SO does not follow the settings of population sizes in the table as it has its own setting (see Section 4.2).

Note that SaDE uses the evaluated individuals to build statistics for parameter control. In this experiment, SaDE in AMALGAM-SO builds this statistic only using the SaDE-generated individuals, which is reasonable. Otherwise, SaDE will no longer be a standalone EA.

To ensure that RandEA and ExhEA share the same statistics as MultiEA for the part which they are common, after executing MultiEA for $N_{\max}$ fitness evaluations, RandEA and ExhEA continue their evolution until each has performed $N_{\max}$ fitness evaluations.

4.5. Simulation results

4.5.1. Simulation 1: comparison with the test multiple-algorithms

In this section, we compare MultiEA with PAP, AMALAGAM-SO, ExhEA, and RandEA. Fig. 7 presents a summary of the results. The shaded cells in the figure indicate that the corresponding test algorithm is the best algorithm on a particular test function. The values inside the cells for MultiEA indicate the ranking of MultiEA on a particular test function when it is not the best algorithm. It can be observed that MultiEA outperforms the other test algorithms. MultiEAs ranks 1st, 2nd, 3rd, 4th and last in 17, 5, 1, 1 and 0 cases, respectively, out of a total of 24 cases.

The detailed simulation results (mean and standard deviation) are listed in Tables A1–A4. It lists the average and the standard deviation (inside brackets) of the best fitness for 100 independent runs. A value in **bold** indicates that the corresponding algorithm is the best amongst the test algorithms on a particular test function. To illustrate the significance of the indicator in Fig. 7, the confidence levels C (in term of %) for Mann–Whitney U test comparing the averaged best fitness values of MultiEA with each of the other algorithms are listed. In the tables, a larger confidence level corresponds to a higher significance in the comparison result.

The procedure for determining significance is as follows: Let there be q algorithms, to determine the significance of the rank of A_j, the significance is tested for the pair (A_j, A_i), for all $i \neq j$. $q-1$ significance values are obtained. The minimum significance value is used to represent the significance of the rank of A_j.

The significance of the rank of MultiEA (i.e., MultiEA ranks 1st in 17, 2nd in 5, 3rd in 1, and 4th in 1 case) is with 99.95% confidence level. Thus, we conclude that the comparisons of MultiEA are statistically significant.

Since MultiEA performs better than RandEA and ExhEA, it suggests that by putting the algorithms in a portfolio using our method, a positive synergy of the algorithm is achieved. Moreover, MultiEA performs better than AMAGLAM-SO and PAP, two state of the art multiple algorithm approaches. This further confirms the power and potential of MultiEA. Experiments have also been conducted for $D = 5$ and 20. Similar ranks of MultiEA are obtained. These results are omitted for brevity.

Observe that the two state of the art methods, AMALGAM-SO and PAP, perform less well than we expected. This suggests that in a self adaptive multiple algorithm approach, the tuning and control of the control parameters in the multiple algorithm may be another tricky problem.

When two algorithms have similar performances (in terms of convergence curves), MultiEA frequently switches between the algorithms and evenly distribute the budget to them. As a result, MultiEA acts like ExhEA in such situations. This may explain why ExhEA sometimes gives better results than expected.

function	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}
MultiEA	1	1	3	1	1	1	1	1	1	2	1	1
PAP	3	2	4	4	2	2	3	3	3	3	3	3
AMALGAM-SO	5	4	2	2	4	4	4	4	5	4	5	4
RandEA	4	5	5	5	5	5	5	5	4	5	4	5
ExhEA	2	3	1	3	3	3	2	2	2	1	2	2

function	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}	f_{21}	f_{22}	f_{23}	f_{24}
MultiEA	1	1	1	2	1	1	4	1	2	2	2	1
PAP	3	3	4	5	3	3	5	4	3	4	5	5
AMALGAM-SO	4	4	3	3	5	5	2	3	4	3	1	2
RandEA	5	5	5	4	4	4	3	5	5	5	4	4
ExhEA	2	2	2	1	2	2	1	2	1	1	3	3

Fig. 7. Indicators of the best test algorithm in the test functions: the cell with gray color represents that the corresponding test algorithm outperforms the others for a particular function.

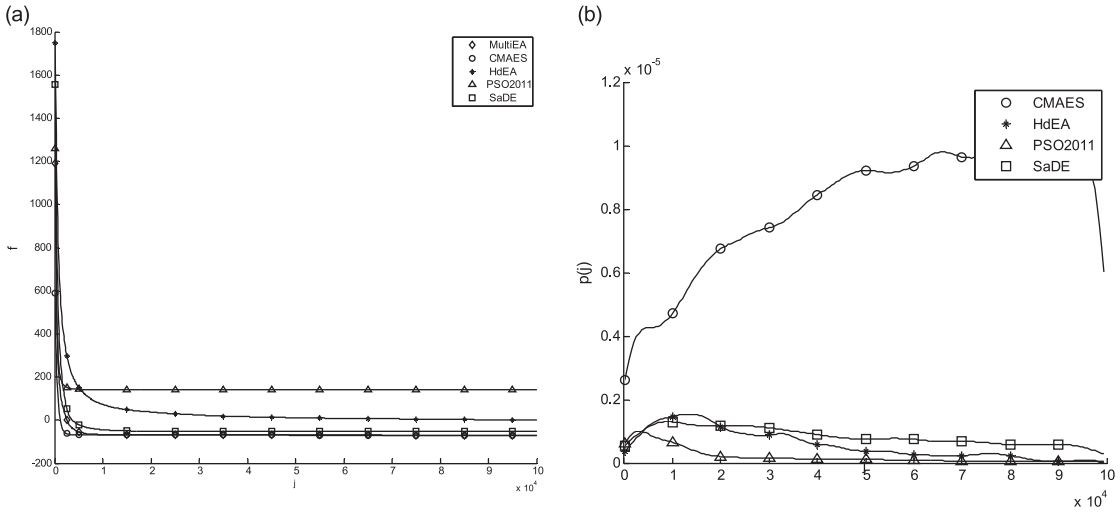

Fig. 8. Function f_{13}: (a) the averaged convergence curves of MultiEA and the other EAs in its portfolio and (b) the frequency $p(j)$ of which EA in the portfolio EA is selected against evaluation index j.

The mechanism of MultiEA can be illustrated through the following two examples. Fig. 8(a) shows the averaged convergence curves of MultiEA and the other EAs in the portfolio on f_{13}. Fig. 8(b) shows the corresponding frequencies of particular EAs chosen during $N_{\max}$ fitness evaluations. Seen from Fig. 8(a), CMA-ES is significantly superior to the other EAs in terms of both convergence rate and the best fitness it can reach. Thus, it is expected that MultiEA provides significantly more budget to CMA-ES than HdEA, PSO2011 and SaDE (Fig. 8(b)).

Fig. 9(a) shows the averaged convergence curves of MultiEA and the other EAs in the portfolio on f_{21}. Fig. 9(b) shows the frequencies of particular EAs chosen. Seen from Fig. 9(b), the budget distribution scheme of MultiEA can be divided into three stages: It initially provides significantly larger amount of budget to CMA-ES. Afterwards, it provides fewer budgets to CMA-ES but more to SaDE and HdEA. In the last stage, the budget for HdEA gradually increases whilst CMA-ES and SaDE get fewer and fewer budgets. This distribution strategy can be understood as follows: In the beginning, as the convergence rate of CMA-ES is the most rapid, more budget is provided for it. However, as the number of fitness evaluations increases, the best fitness of CMA-ES remains at around 324 but those of SaDE and HdEA keep improving at different rates. Hence MultiEA pays more attention on SaDE and HdEA. In the latter stage of the search process, as HdEA keeps improving its best solution, relative more budgets are provided to it. Concluding from these two examples, MultiEA would choose a "fast converging" algorithm for a small budget. If more budgets are provided, it turns to explore a "more globally converging" algorithm.

4.5.2. Simulation 2: neck to neck comparisons with evolutionary algorithms in the portfolio

Fig. 10 presents a summary of the comparison results of MultiEA and the four EAs in the portfolio. The shaded cells in the figure indicate that the corresponding test algorithm is the best algorithm on a particular test function. The values inside the table cells for MultiEA indicate the ranking of MultiEA on a particular test function when it is not the best algorithm.

Recall that the goal of MultiEA is to identify the *best* EA in the portfolio as quickly as possible. Thus 1st rank is not a reasonable requirement to show the superiority of MultiEA; it is not expected to beat the algorithm which it finds to be the best compared with the rest, as it takes time to discover that algorithm. Instead, 2nd rank is a reasonable expectation sufficient to demonstrate the effectiveness. The detailed simulation results (mean and standard deviation) are listed in Tables B1–B4. Fig. 11 shows the histogram of the ranks of MultiEA and the four EAs in portfolio. Seen from the figure, MultiEA ranks 1st and 2nd in 5 and 13 out of a total of 24 cases, respectively. Except for f_1 that the significance of the rank of MultiEA is with lower 50% confidence, the significance of the rank of MultiEA for all other functions is with 99.95% confidence. The lowest rank that MultiEA has is 4, which occurs in one case; this shows that it maintains a relatively good performance when it is not the best algorithm.

4.5.3. A positive synergic effect

Remarkably, MultiEA ranks 1st in 5 out of 24 cases, which exceeds our theoretical expectation. This surprising result can be

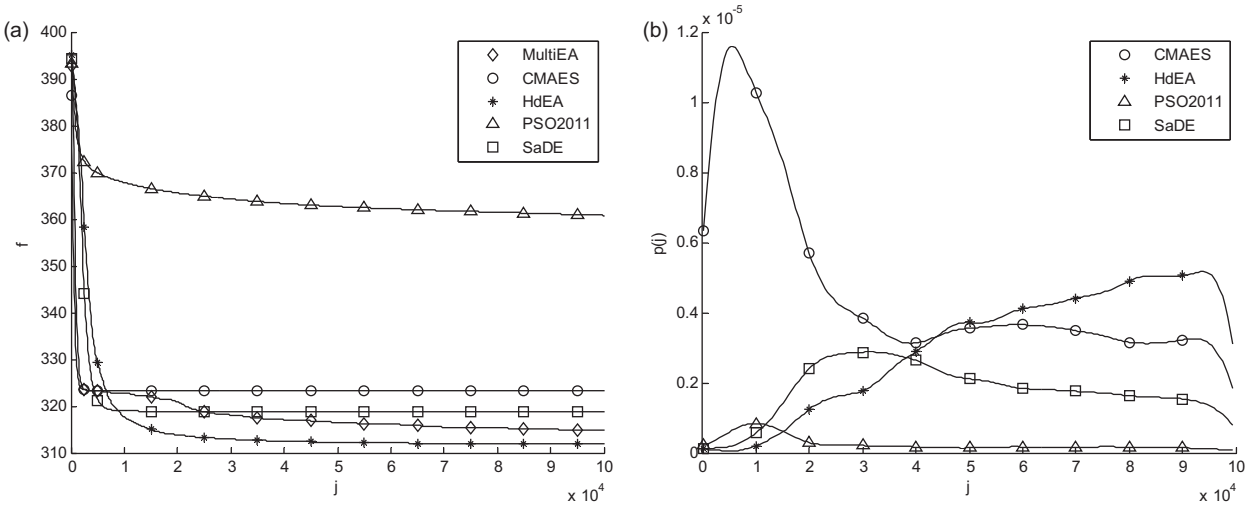

Fig. 9. Function f_{21}: (a) the averaged convergence curves of MultiEA and the other EAs in its portfolio and (b) the frequency $p(j)$ of which EA in the portfolio is selected against evaluation index j.

function	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}
MultiEA	1	4	2	2	3	2	2	3	2	2	3	1
CMA-ES	1	5	4	4	4	5	1	1	1	1	4	4
HdEA	4	3	1	1	2	3	4	2	3	4	5	3
PSO2011	5	1	5	5	5	4	5	5	5	5	1	5
SaDE	3	1	3	3	1	1	3	4	4	3	2	2

function	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}	f_{21}	f_{22}	f_{23}	f_{24}
MultiEA	1	2	2	3	1	1	3	2	2	2	2	2
CMA-ES	2	1	1	1	2	2	1	4	4	4	3	1
HdEA	4	4	4	2	5	5	5	1	1	1	1	5
PSO2011	5	5	5	4	4	4	2	5	5	5	4	4
SaDE	3	3	3	5	3	3	4	3	3	3	5	3

Fig. 10. Indicators of the best test algorithm in the test functions: the cell with gray color represents that the corresponding test algorithm outperforms the others for a particular function.

explained as follows: Since all the selected EAs are stochastic, it may happen that the average performance of algorithm A is better than that of algorithm B, but in some runs B is better than A. In the case in which the performance of A and B are comparable but not stable, the proposed algorithm selection scheme of MultiEA may assign more effort to the better algorithm at a particular run: for a run that A performs better than B, MultiEA prefers executing A. Alternatively, for a run that B performs better than A, MultiEA prefers executing B. As a result, the average performance of MultiEA would be better than *both* A and B.

The above clearly indicates that MultiEA combines different EAs and creates a positive synergic effect. By this we mean that by combining two algorithms A_1 and A_2 in our portfolio, the resultant algorithm may be one which is better than both. Literally, it embodies the sayings "one plus one is larger than two" or "the whole is larger than the sum of parts".

This effect can be illustrated through the experimental results on f_{17}, which is one of the four functions of which MultiEA ranks 1st. We record the best fitness values of MultiEA as well as the four EAs in the portfolio at the Nth fitness evaluation of each run. Fig. 12 shows the probability density functions of the recorded best fitness of the five algorithms. Let an algorithm at a particular run be classified as *good* if the corresponding best fitness is lower than −51. In the 35 runs that CMA-ES is *good*, most of the best fitness found by MultiEA also fall in the range [−52, −51] but those of SaDE is sparsely distributed in a wider range from −52 to −50 (Fig. 12(b)). Meanwhile, for the 38 runs that SaDE is *good*, the best fitness found by MultiEA are also mainly in the range [−52, −51] but those of CMA-ES is sparsely distributed in a wider range from −52 to −47 (Fig. 12(c)). Concluding from these observations, the performance of CMA-ES and SaDE on f_{17} is quite extreme and opposite to each other. On the other hand, MultiEA can trace the best algorithm at each particular run. The stable performance of MultiEA is the reason why it obtains the 1st rank in the average performance measure.

4.5.4. Simulation 3: scalability of MultiEA

Conventionally, a scalability analysis is an analysis of the dimensionality against the average number of function evaluations (with uncertainty estimates) required to reach a user defined convergence criterion. However, it is important to note that such a scalability analysis will not yield any additional insight to MultiEA, as the scalability of MultiEA clearly depends on what individual algorithms are chosen to compose the portfolio. This observation motivates us to do the scalability analysis below, which attempts to investigate the scaling behavior of MultiEA in comparison with the best algorithm within the portfolio. The procedure is as follows:

We experimentally study the scalability of MultiEA by investigating the best fitness found by MultiEA against function dimension. The examined dimensions are D = 40, 80, 100, 120, 150, 180 and 200. We repeat the process at Section 4.5.2 for different dimensions. Fig. 13 shows the averaged best fitness f against number of dimensions on four test functions: f_1, f_5, f_{15} and f_{24}. Note that since the scalability of PSO2011 on f_1 is not as good as those of other studied algorithms, the y-axis in Fig. 13(a) is nonlinearly divided into two ranges [−100, −70] and [−30, 930] to show the curves of all studied algorithms. As the scalability of MultiEA is bounded by

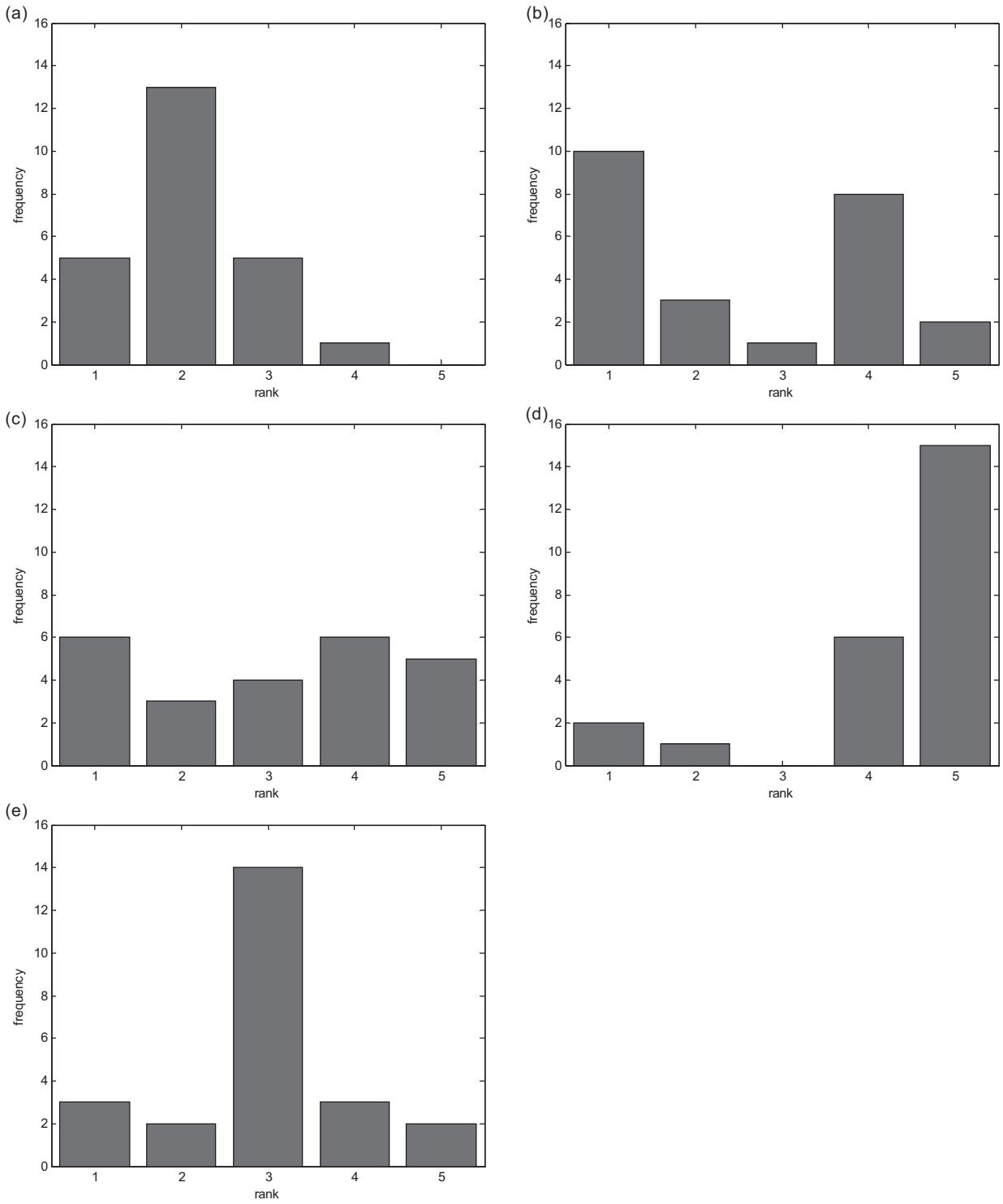

Fig. 11. Histograms of the ranks of (a) MultiEA, (b) CMA-ES, (c) HdEA, (d) PSO2011 and (e) SaDE on the 24 test cases.

those of the EAs in the portfolio, we should focus on how close MultiEA is to the scalability of the best EA in the portfolio, rather than the exact order of its scalability. Seen from the figure, MultiEA has good scalability as its best fitness is quite close to that of the best EA in the portfolio across all examined D.

4.5.5. Simulation 4: application to a real world problem

In this section, we apply MultiEA to the design of a HVAC (Heating, Ventilating, and Air Conditioning) system. This real world application is a duct design minimization problem [35] with 19 variables ($D = 19$). The total life cycle cost is to be minimized. The problem has 88 constraints. The selection operator is modified for constraint handling. Following [35], we use Deb's constraint handling method [36], which is based on three rules:

1. Any feasible solution is preferred to any infeasible solution.
2. For two feasible solutions, the one with the smaller function value is preferred.

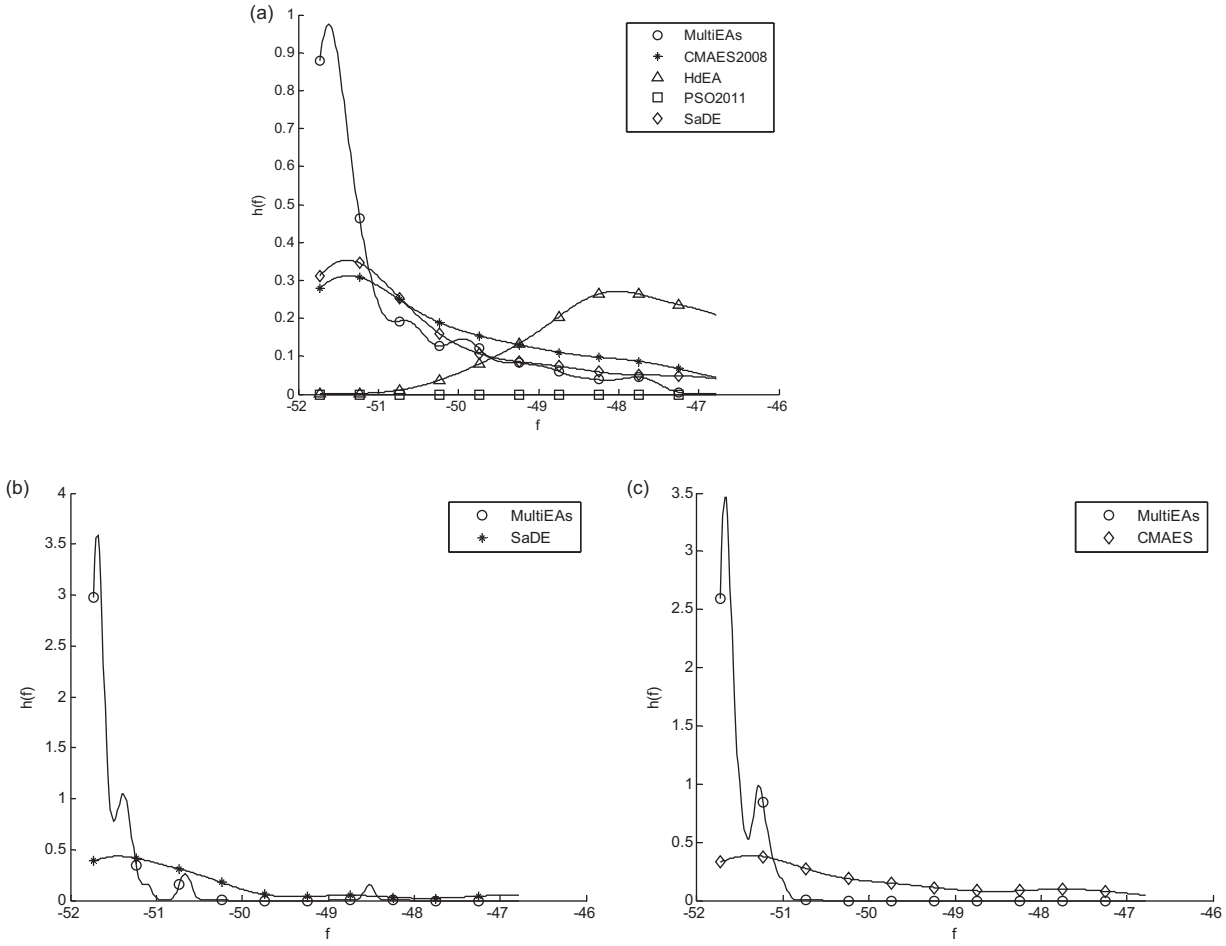

Fig. 12. (a) The probability density functions $h(f)$ of the best fitness of f_{17} found by MultiEA and the four EAs in the portfolio, (b) the probability density functions $h(f)$ of best fitness of f_{17} found by MultiEA and SaDE for the runs that the best fitness found by CMA-ES is lower than −51, and (c) the probability density functions $h(f)$ of best fitness of f_{17} found by MultiEA and CMA-ES for the runs that the best fitness found by SaDE is lower than −51.

3. For two infeasible solutions, the fitness value and the weighted constraint violation penalty value is added. The solution with the smaller total value is preferred. The weighting is set to 1 in this paper.

Since [35] is a design problem, the best result in a number of runs is used to measure the performance. Please refer to [35] for more details about the problem.

In this experiment, the population sizes of CMA-ES, HdEA, PSO 2011, SaDE are set to 12, 20, 19 and 50, respectively, following the recommended settings in the literature [3–5,7].

MultiEA are compared with the baseline RandEA, as well as individual algorithms in MultiEA, i.e., CMA-ES, HdEA, PSO2011 and SaDE. The maximum number of evaluations is set to 2000. 20 independent runs are performed by each algorithm. Fig. 14 shows the best results in 20 runs. The total cost, which is the sum of the material and energy costs, is to be minimized.

CMA-ES ranks 1st and is the best algorithm for this problem. Since we have no a priori knowledge about which algorithm is the best, MultiEA takes time to discover that CMA-ES is the best. Thus it ranks 2nd. Note also that RandEA ranks 6th and is the worst. This means that one cannot obtain good results if we have had chosen an algorithm randomly. The experimental results suggest that given an unknown problem, MultiEA is effective in choosing the best algorithm for the problem. It also shows that MultiEA is effective in solving a real world application problem with constraints.

5. Conclusions

5.1. Summary of contributions

Evolutionary algorithms (EAs) have proliferated as one of the best tools for solving the ubiquitous optimization problem in the real world. Many excellent EAs have been reported in the past. However, there arises an important open problem which bewilders researchers and engineers alike: Confronted with an optimization problem, *which algorithm should I choose*?

This paper is an attempt to answer the above question. A novel algorithm, known as Multiple Evolutionary Algorithm (MultiEA), is proposed for single objective optimization. A portfolio is first formed by selecting several state of the art EAs. A novel predictive measure is reported to predict the performance of individual algorithms if they were extrapolated to the *same* number of evaluations in the nearest future. The algorithm with the best predicted performance is chosen to run for one generation. New search information is received and the history is updated. The predicted performance of the algorithms is updated and the algorithm with the best predicted performance is re-selected, which may or may not be the same algorithm in the last generation. Experimental results show that the measure is stable and is a reasonably effective predictor. The idea is simple and natural. It is parameter-less. It does not introduce any new control parameter to the algorithm, thus avoiding the challenging parameter tuning and control problem [2].

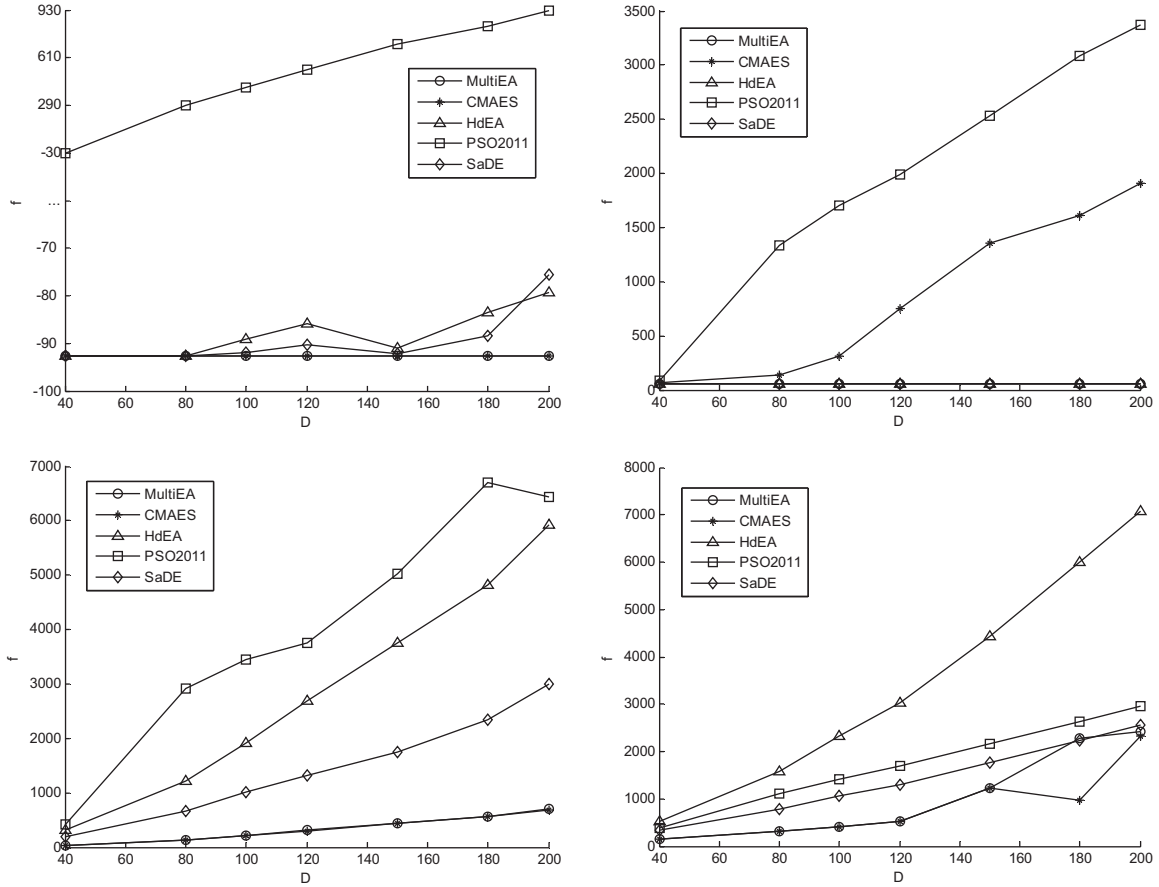

Fig. 13. Scalability analysis of MultiEA: the averaged best fitness f against number of dimensions D. (a) f_1, (b) f_5, (c) f_{15} and (d) f_{24}. The scalability of PSO2011 on f_1 is not as good as those of other studied algorithms. In order to show the curves of all studied algorithms in (a), the y-axis is nonlinearly divided into two ranges [−100, −70] and [−30, 930].

	MultiEA	RandEA	HdEA	SaDE	PSO 2011	CMA-ES
Rank	2	6	4	3	5	1
Total cost, $	11227	11686	11281	11254	11678	11216
Material cost, $	8098	8498	8300	8190	8296	8060
Energy cost, $	3129	3188	2981	3064	3382	3156
x1	29	36	32	31	28	31
x2	20	20	21	21	20	21
x3	41	43	44	41	41	39
x4	60	60	60	60	60	60
x5	46	47	51	48	45	46
x6	68	59	64	67	66	68
x7	26	64	21	32	38	28
x8	51	27	29	50	49	49
x9	43	54	44	38	34	34
x10	48	56	50	49	54	48
x11	30	63	34	31	51	31
x12	30	27	33	31	28	30
x13	35	43	33	35	38	33
x14	61	68	77	67	62	67
x15	18	20	19	18	27	17
x16	19	25	20	20	34	18
x17	31	31	31	32	54	33
x18	80	67	77	76	58	79
x19	80	80	80	80	80	80

Fig. 14. Comparison of the results of MultiEA, RandEA, HdEA, SaDE, PSO 2011, and CMA-ES for the duct system design problem.

Table A1
The mean, standard deviation (stdev.) and confidence level (C) of the best fitness values found by MultiEA, PAP, AMALGAM-SO, RandEA and ExhEA: f_1–f_6.

		f_1	f_2	f_3	f_4	f_5	f_6
MultiEA	Mean	**−92.65**	**276.32**	26.801279	**26.387526**	**51.53**	**83.560103**
	Stdev.	**(4.86280E−6)**	**(3.81470E−6)**	(19.8141)	**(24.3724)**	**(1.78416E−6)**	**(1.60658E−1)**
PAP	Mean	−92.617308	276.32000	96.490389	115.34190	51.53	86.237335
	Stdev.	(2.05818E−2)	(9.34406E−6)	(11.2120)	(13.3706)	(4.42201E−6)	(2.00136)
	C	99.95%	99.95%	99.95%	99.95%	99.95%	99.95%
AMALGAM-SO	Mean	−20.495780	280.04575	26.662447	33.775714	51.647992	169.88709
	Stdev.	(28.3819)	(4.13673)	(3.20858)	(6.66513)	(5.99650E−2)	(61.9279)
	C	99.95%	99.95%	99.95%	99.95%	99.95%	99.95%
RandEA	Mean	−74.460936	8707.2872	189.49731	248.33533	62.263728	7207.6557
	Stdev.	(3.81677)	(59,674.4)	(21.8948)	(21.8354)	(10.3032)	(4851.73)
	C	99.95%	99.95%	99.95%	99.95%	99.95%	99.95%
ExhEA	Mean	−92.64999	276.32000	**26.239301**	33.865407	51.530018	92.768933
	Stdev.	(2.57E−8)	(1.07896E−5)	**(1.47984)**	(2.33822)	(1.56995E−5)	(7.25155)
	C	99.95%	99.95%	**99.95%**	99.95%	99.95%	99.95%

Table A2
The mean, standard deviation (stdev.) and confidence level (C) of the best fitness values found by MultiEA, PAP, AMALGAM-SO, RandEA and ExhEA: f_7–f_{12}.

		f_7	f_8	f_9	f_{10}	f_{11}	f_{12}
MultiEA	Mean	**−70.665692**	**−128.69591**	**−354.06604**	794.14020	**−33.721796**	**295.43627**
	Stdev.	**(5.26607)**	**(8.89037)**	**(5.21592)**	(2402.45)	**(41.1373)**	**(1.08910)**
PAP	Mean	−52.581774	−53.753867	−318.14736	16,296.658	−3.6873870	299.04314
	Stdev.	(9.17643)	(32.8707)	(15.0267)	(6145.02)	(18.2248)	(4.88755)
	C	99.95%	99.95%	99.95%	99.95%	99.95%	99.95%
AMALGAM-SO	Mean	4.8682347	7.3417433	−1.8227026	34,442.592	78.538467	11,620.135
	Stdev.	(13.9319)	(2.04773)	(7.93527)	(12,105.9)	(37.6881)	(3133.54)
	C	99.95%	99.95%	99.95%	99.95%	99.95%	99.95%
RandEA	Mean	4.2859759	5327.4259	−180.74968	66,306.419	42.102655	2.7226564E+7
	Stdev.	(14.2489)	(2057.38)	(56.9368)	(20,344.5)	(45.6257)	(9.26122E+6)
	C	99.95%	99.95%	99.95%	99.95%	99.95%	99.95%
ExhEA	Mean	−68.289700	−107.03428	−335.58107	**383.20397**	−8.5134107	295.50474
	Stdev.	(5.73896)	(14.2225)	(2.38281)	**(315.465)**	(27.7630)	(1.09731)
	C	99.95%	99.95%	99.95%	**99.95%**	99.95%	99.95%

Table A3
The mean, standard deviation (stdev.) and confidence level (C) of the best fitness values found by MultiEA, PAP, AMALGAM-SO, RandEA and ExhEA: f_{13}–f_{18}.

		f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}
MultiEA	Mean	**−50.875884**	**−57.899998**	**26.440962**	−245.81029	**−38.663591**	**−38.212070**
	Stdev.	**(1.20599)**	**(6.74350E−7)**	**(21.5359)**	(2.96937)	**(6.67686E−2)**	**(3.28783E−1)**
PAP	Mean	−49.158216	−57.895822	218.30133	−233.82848	−38.160676	−35.760863
	Stdev.	(2.81590)	(8.10509E−4)	(19.4700)	(2.28644)	(2.94323E−1)	(1.39320)
	C	99.95%	99.95%	99.95%	99.95%	99.95%	99.95%
AMALGAM-SO	Mean	−2.4276550E−1	−57.323187	197.47491	−244.46618	−35.364175	−21.235029
	Stdev.	(3.46812E−1)	(5.99350E−1)	(50.8560)	(4.65375)	(6.04847E−1)	(5.59187)
	C	99.95%	99.95%	99.95%	99.95%	99.95%	99.95%
RandEA	Mean	282.73916	−54.841512	238.07353	−245.04779	−35.564037	−27.200867
	Stdev.	(43.7993)	(6.47425E−1)	(22.3046)	(3.06217)	(3.58372E−1)	(1.08741)
	C	99.95%	99.95%	99.95%	99.95%	99.95%	99.95%
ExhEA	Mean	−50.081474	−57.899963	25.296135	**−253.76800**	−38.660491	−38.205096
	Stdev.	(1.51122)	(6.91003E−6)	(16.0483)	**(7.54378)**	(7.89741E−2)	(3.41574E−1)
	C	99.95%	99.95%	99.95%	**99.95%**	99.95%	99.95%

Each component algorithm retains and uses their recommended set of parameters, which is the standard practice in the evolutionary community. No parameter tuning and control is required. Instead, our approach expects individual algorithms to play complementary roles. Different algorithms which excel in different problems will stand out when required. Moreover, as different algorithms may be the best for different computational budgets in the sense of absolute maximum number of fitness evaluations, the approach fully allows the selection of the best algorithm given a fixed computational budget, as well as automatic algorithm switching as the budget varies.

Some recent multiple algorithm portfolio approaches use a common population and a self adaptive approach to apportion different algorithms at different stages. Compared with these approaches, ours use a distinctly different philosophy, namely, we concentrate on selecting the best algorithm given the current computational budget and predicted performance. We believe that these two approaches are complementary, and this paper provides a fresh alternative.

A novel performance evaluation measure is also proposed. In many previous papers, the algorithms are compared after a fixed number of evaluations. This is not very fair as some algorithms may

Table A4
The mean, standard deviation (stdev.) and confidence level (*C*) of the best fitness values found by MultiEA, PAP, AMALGAM-SO, RandEA and ExhEA: f_{19}–f_{24}.

		f_{19}	f_{20}	f_{21}	f_{22}	f_{23}	f_{24}
MultiEA	Mean	45.121719	**183.88420**	314.80036	45.762437	212.87839	**183.33640**
	Stdev.	(7.97964E−1)	**(4.61367E−1)**	(3.94903)	(1.90927)	(9.04757E−1)	**(78.1294)**
PAP	Mean	46.466411	185.55870	318.72193	46.338104	213.97604	358.96465
	Stdev.	(3.02763E−1)	(2.07738E−1)	(3.44890)	(3.76024)	(3.84971E−1)	(19.2574)
	C	99.95%	99.95%	99.95%	99.95%	99.95%	99.95%
AMALGAM-SO	Mean	45.042856	184.16945	319.28870	46.338204	**212.41808**	230.15338
	Stdev.	(1.12397)	(1.58794E−1)	(5.28651)	(3.76036)	**(7.89777E−1)**	(31.9548)
	C	99.95%	99.95%	99.95%	99.95%	**99.95%**	99.95%
RandEA	Mean	44.961906	226.97935	328.74557	58.219278	213.01249	330.55127
	Stdev.	(2.29008E−1)	(78.1148)	(3.66256)	(3.26167)	(3.68984E−1)	(16.6542)
	C	99.95%	99.95%	99.95%	99.95%	99.95%	99.95%
ExhEA	Mean	**44.412577**	184.14202	**313.09923**	**45.570615**	212.91794	216.66614
	Stdev.	**(1.81021)**	(8.23445E−2)	**(1.69516)**	**(6.67226E−4)**	(4.19954E−1)	(98.2070)
	C	**99.95%**	99.95%	**99.95%**	**99.95%**	99.95%	99.95%

Table B1
The mean, standard deviation (stdev.) and confidence level (*C*) of the best fitness values found by MultiEA, CMAES, HdEA, PSO2011 and SaDE: f_1–f_6.

		f_1	f_2	f_3	f_4	f_5	f_6
MultiEA	Mean	**−92.65**	276.32712	27.350188	26.451605	52.688561	83.560341
	Stdev.	**(3.00000E−8)**	(5.29300E−2)	(20.3303)	(24.8676)	(8.42275)	(1.61804E−1)
CMAES	Mean	**−92.65**	34,008.090	91.551954	138.54358	64.720252	17,785.145
	Stdev.	**(00000)**	(2.39899E+5)	(17.1307)	(29.5465)	(6.80084)	(14,360.3)
	C	**< 50%**	99.95%	99.95%	99.95%	99.95%	99.95%
HdEA	Mean	−92.649690	276.32008	**20.91**	**20.91**	51.538538	159.95640
	Stdev.	(2.52947E−3)	(1.48780E−4)	**(00000)**	**(1.00000E−8)**	(3.28961E−2)	(28.2179)
	C	99.95%	99.95%	**99.95%**	**99.95%**	99.95%	99.95%
PSO2011	Mean	−21.403372	**276.32000**	614.29909	785.84608	118.73258	10,804.452
	Stdev.	(15.2280)	**(00000)**	(84.5040)	(81.2427)	(40.7923)	(11,116.3)
	C	99.95%	**99.95%**	99.95%	99.95%	99.95%	99.95%
SaDE	Mean	−92.649999	**276.32000**	32.348187	48.070038	**51.530397**	**83.491160**
	Stdev.	(8.56000E−6)	**(00000)**	(5.44242)	(6.81418)	**(1.96095E−3)**	**(2.11298E−2)**
	C	99.95%	**99.95%**	99.95%	99.95%	**99.95%**	**99.95%**

Table B2
The mean, standard deviation (stdev.) and confidence level (*C*) of the best fitness values found by MultiEA, CMAES, HdEA, PSO2011 and SaDE: f_7–f_{12}.

		f_7	f_8	f_9	f_{10}	f_{11}	f_{12}
MultiEA	Mean	−70.337816	−119.76170	−348.45575	983.12773	−19.141360	**295.46683**
	Stdev.	(5.31149)	(17.4829)	(4.58309)	(2838.02)	(43.5956)	**(1.08890)**
CMAES	Mean	**−72.102537**	**−134.69147**	**−359.28054**	**−72.709344**	47.882360	6,697,515.8
	Stdev.	**(4.29607)**	**(1.25366)**	**(7.85151E−1)**	**(50.6729)**	(177.136)	(2.49385E+7)
	C	**99.95%**	**99.95%**	**99.95%**	**99.95%**	99.95%	99.95%
HdEA	Mean	5.1238844	−128.30104	−322.48554	93,060.408	213.84038	912.13716
	Stdev.	(33.8950)	(9.40598)	(1.03612)	(21,639.7)	(38.9821)	(594.547)
	C	99.95%	99.95%	99.95%	99.95%	99.95%	99.95%
PSO2011	Mean	140.96218	21,857.298	291.27685	175,089.43	**−28.267891**	1.0500746E+8
	Stdev.	(53.8275)	(8357.31)	(229.320)	(81,233.7)	**(28.6670)**	(2.61379E+7)
	C	99.95%	99.95%	99.95%	99.95%	**99.95%**	99.95%
SaDE	Mean	−51.267259	−70.853691	−320.82337	6825.0418	−25.422489	299.35233
	Stdev.	(11.2897)	(31.0920)	(11.5943)	(3034.85)	(22.6014)	(4.37736)
	C	99.95%	99.95%	99.95%	99.95%	99.95%	99.95%

have quickly converged to the target fitness and they should be declared superior to slower algorithms. Our measure overcomes this limitation by dynamically changing the number of evaluations on each run based on the first to target concept and an absolute maximum computational budget limit concept. The measure is general and can be applied to other EA and non-EA comparison context.

Four algorithms, namely, Covariance Matrix Adaptation Evolution Strategy (CMA-ES), History driven Evolutionary Algorithm (HdEA), Particle Swarm Optimization (PSO2011), and Self adaptive Differential Evolution (SaDE) are selected to compose our portfolio in MultiEA in our experiments. These algorithms are selected because they are state of the art algorithms, and have significantly different characteristics, strengths and pitfalls.

Experimental results on 24 benchmark functions show that MultiEA outperforms (i) Multialgorithm Genetically Adaptive Method for Single-Objective Optimization (AMALGAM-SO); (ii) Population-based Algorithm Portfolio (PAP); (iii) a simple multiple algorithm

Table B3
The mean, standard deviation (stdev.) and confidence level (*C*) of the best fitness values found by MultiEA, CMAES, HdEA, PSO2011 and SaDE: f_{13}–f_{18}.

		f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}
MultiEA	Mean	**−50.875850**	−57.899998	32.539355	−244.66332	**−38.663591**	**−38.212069**
	Stdev.	**(1.21205)**	(8.40000E−7)	(46.7859)	(3.44854)	**(6.71050E−2)**	**(3.30439E−1)**
CMAES	Mean	−50.154728	**−57.9**	**25.297363**	**−253.42606**	−38.585282	−38.197194
	Stdev.	(1.66993)	**(00000)**	**(16.1291)**	**(11.5915)**	(7.99279E−1)	(3.65440E−1)
	C	99.95%	**99.95%**	**99.95%**	**99.95%**	99.95%	99.95%
HdEA	Mean	−39.488217	−57.887519	360.20111	−244.89486	−31.560428	−12.750154
	Stdev.	(24.3998)	(1.69567E−3)	(65.3320)	(2.39339)	(9.73556E−1)	(3.55219)
	C	99.95%	99.95%	99.95%	99.95%	99.95%	99.95%
PSO2011	Mean	1286.3345	−45.679274	459.71291	−241.22508	−33.596692	−20.478407
	Stdev.	(179.296)	(2.60309)	(66.3392)	(3.36606)	(6.59423E−1)	(2.60009)
	C	99.95%	99.95%	99.95%	99.95%	99.95%	99.95%
SaDE	Mean	−48.707765	−57.899087	202.42024	−235.04330	−38.202162	−36.459607
	Stdev.	(4.88510)	(1.47130E−4)	(31.2669)	(2.86348)	(2.88400E−1)	(1.05580)
	C	99.95%	99.95%	99.95%	99.95%	99.95%	99.95%

Table B4
The mean, standard deviation (stdev.) and confidence level (*C*) of the best fitness values found by MultiEA, CMAES, HdEA, PSO2011 and SaDE: f_{19}–f_{24}.

		f_{19}	f_{20}	f_{21}	f_{22}	f_{23}	f_{24}
MultiEA	Mean	45.125804	183.88803	314.88547	46.305009	212.89766	188.17803
	Stdev.	(8.01498E−1)	(4.64453E−1)	(3.95854)	(3.49553)	(9.10383E−1)	(86.4602)
CMAES	Mean	**41.425602**	185.00035	323.39550	49.480375	213.02043	**146.68117**
	Stdev.	**(2.96008E−1)**	(1.81316E−1)	(10.1235)	(7.36481)	(1.32853)	**(23.7893)**
	C	**99.95%**	99.95%	99.95%	99.95%	99.95%	**99.95%**
HdEA	Mean	47.315753	**183.58869**	**312.22318**	**45.985160**	**212.02522**	511.83724
	Stdev.	(4.90725E−1)	**(7.36778E−2)**	**(1.97290)**	**(9.26661E−1)**	**(3.04874E−1)**	(48.4619)
	C	99.95%	**99.95%**	**99.95%**	**99.95%**	**99.95%**	99.95%
PSO2011	Mean	45.083829	354.67778	361.12904	94.704481	213.44032	389.61174
	Stdev.	(5.13437E−1)	(314.149)	(10.9619)	(10.1532)	(3.67173E−1)	(34.6454)
	C	99.95%	99.95%	99.95%	99.95%	99.95%	99.95%
SaDE	Mean	46.050943	184.84017	318.95158	47.311004	213.63407	336.32536
	Stdev.	(3.21451E−1)	(4.50975E−1)	(3.82185)	(5.53281)	(3.59649E−1)	(22.2906)
	C	99.95%	99.95%	99.95%	99.95%	99.95%	99.95%

approach which chooses an algorithm randomly (RandEA); and (iv) a simple multiple algorithm approach which divides the computational budget evenly and execute all algorithms in parallel (ExhEA). This shows that MultiEA outperforms existing portfolio approaches and the predictor is functioning well.

A more stringent test is also performed by comparing MultiEA with the individual algorithms (i.e., CMA-ES, HdEA, PSO2011, SaDE) in a neck to neck manner. It is found that MultiEA is highly competitive. In the majority of cases, it ranks second, which is to be fully expected as it does have to spend some budget to identify the best algorithm within the portfolio. This means that the predictive measure that we propose is doing a competent job. Also, it suggests that MultiEA is competent as a novel standalone EA. In particular, MultiEA, being a portfolio algorithm, is sometimes even better than all its individual algorithms, and has more robust performance.

Surprisingly, we find that for some cases, MultiEA ranks first, which leads to the discovery of an interesting positive synergic effect. This shows that the portfolio strategy of MultiEA can sometimes outperform even the best individual algorithm. Literally, "one plus one is larger than two," or "the whole is larger than the sum of parts". This gives interesting insights into why a portfolio approach is a good approach.

A good algorithm should scale up nicely. A scalability test is also performed for MultiEA. Realistically, one should be content that MultiEA scales as good as the best performing component EA in the portfolio. We found that it is indeed the case.

MultiEA is also applied to a real world application in Heating Ventilation and Air Conditioning (HVAC) engineering [35]. It is found that MultiEA can determine the most suitable algorithm in the portfolio for the problem, and the performance is much better than randomly choosing an algorithm. It also demonstrates that MultiEA is effective in problems involving constraints.

Though the scope of this paper is restricted to portfolio of EAs, it is a general framework and good non-EA (e.g. Efficient Global Optimization algorithm (EGO) [37]) is also welcome.

Theoretically, MultiEA, by virtue that it is just another EA, will still be under the no free lunch (NFL) theorems [11]. There are likely to be situations that MultiEA gets a bad result. However, the premise of the NFL is that all computable problems are equally likely to occur. It is hardly the case in the real world [38]. In fact, the promise of multi-method search is that by using many search metaphors inspired by nature, mathematics, sciences, arts, etc., one would build up better search strategies that are more able to deal with real world problems. It may perhaps be likened to synergy of individuals with different talents in a heterogeneous society.

5.2. Limitations and future works

In this paper, we have used our personal expertise and judgment to select algorithms to compose the portfolio. In principle, if we have had selected a very bad algorithm, it would not affect the

performance of MultiEA too much. This is because MultiEA will only select the best algorithm to run at any one time.

Recently, a systematic and principled approach to solve this problem has been developed to solve this problem [25–27]. Given a set of (evolutionary or non-evolutionary) algorithms and a set of existing problems that have been encountered, the methods in [25–27] may be used to compose a suitable portfolio that performs well on average. Such research complements this research nicely.

Another interesting future research is how to design algorithms with restarts in a portfolio. In the first restart onwards, the predictor may make use of information in the previous runs to arrive at better prediction. The pros and cons of making use of such information deserve further study.

We have not considered the overheads due to solution generation costs. In many practical optimization, the fitness evaluation costs is much greater than the solution generation costs and the latter can be ignored. However, in the case that the solution generation costs are substantial, this factor needs to be taken into account. Different algorithms may have different solution generation costs due to its different algorithm complexity. For example, the solution generation cost of EGO [37] increases nonlinearly with the number of evaluations. Since the overall computational budget for the problem is fixed, a more sophisticated predictive measure will take into account of the differing solution costs when suggesting the most suitable algorithm at any particular point.

In this work, we have concentrated on single objective optimization problem for which the fitness function is single-valued and has no uncertainty. While this is true for some real world applications, there are other applications that involve more complex considerations. As discussed in [39,40], (1) real world applications frequently involve multiple, potentially conflicting objectives; (2) their fitness functions may have time varying uncertainties arising out of different factors; (3) sometimes robust designs insensitive to parameter changes are required rather than optimized designs; (4) some fitness functions may have qualitative components or may require interactions with humans to obtain a meaningful values. Extensions and adaptation of MultiEA are required to tackle the issues above.

Future research may also be done on developing better prediction measures, which predicts more accurately which algorithm should be applied at the nearest common future point.

The initial motivation of this paper is a question from a colleague: there are so many evolutionary algorithms claimed to be powerful and useful, which algorithm should I choose? As this question comes from a user of evolutionary computation for optimization of practical problems, we hope that our paper has answered him in one way. We encourage more research in this interesting and important direction.

Acknowledgments

The question studied in this paper was raised by Ron S.Y. Hui, a researcher interested in applying evolutionary algorithms to energy and environmental applications, during a causal discussion with the first author. The work described in this paper was supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China [Project No. CityU 125313]. Yang Lou was also funded by a Research Studentship from City University of Hong Kong.

References

[1] A.P. Engelbrecht, Computational Intelligence: An Introduction, second ed., Wiley Publishing, 2007.
[2] G. Karafotias, M. Hoogendoorn, A.E. Eiben, Parameter control in evolutionary algorithms: trends and challenges, IEEE Trans. Evol. Comput. 19 (2) (2015) 167–187.
[3] N. Hansen, The CMA Evolution Strategy: A Tutorial, Technical Report, 28 June, 2011, URL: http://www.lri.fr/~hansen/cmatutorial.pdf.
[4] C.K. Chow, S.Y. Yuen, An evolutionary algorithm that makes decision based on the entire previous search history, IEEE Trans. Evol. Comput. 15 (6) (2011) 741–769.
[5] Particle Swarm Central. URL: http://www.particleswarm.info/.
[6] S. Das, P.N. Suganthan, Differential evolution: a survey of the state-of-the-art, IEEE Trans. Evol. Comput. 15 (February (1)) (2011) 4–31.
[7] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evol. Comput. 13 (April (2)) (2009) 398–417.
[8] A.M. Sutton, M. Lunacek, L.D. Whitley, Differential evolution and non-separability: using selective pressure to focus search, in: Proc. of the Genetic and Evolutionary Computation Conference, 2007, pp. 1428–1435.
[9] M. Preuss, G. Rudolph, S. Wessing, Tuning optimization algorithms for real-world problems by means of surrogate modeling, in: Proc. of the Genetic and Evolutionary Computation Conference, 2010, pp. 401–408.
[10] W.G. Macready, D.H. Wolpert, What makes an optimization problem hard? Complexity 5 (1) (1996) 40–46.
[11] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (April (1)) (1997) 67–82.
[12] Black-Box Optimization Benchmarking (BBOB) 2010, URL: http://coco.gforge.inria.fr/doku.php?id=bbob-2010, 2010.
[13] J.A. Vrugt, B.A. Robinson, J.M. Hyman, Self-adaptive multimethod search for global optimization in real-parameter spaces, IEEE Trans. Evol. Comput. 13 (April (2)) (2009) 243–259.
[14] F. Peng, K. Tang, G. Chen, X. Yao, Population-based algorithm portfolios for numerical optimization, IEEE Trans. Evol. Comput. 14 (October (5)) (2010) 782–800.
[15] E.K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, R. Qu, Hyper-heuristics: a survey of the state of the art, J. Oper. Res. Soc. 64 (12) (2013) 1695–1724.
[16] B.A. Huberman, R.M. Lukose, T. Hogg, An economics approach to hard computational problems, Science 275 (January) (1997) 51–54.
[17] J. Rachlin, R. Goodwin, S. Murthy, R. Akkiraju, F. Wu, S. Kumaran, R. Das, A-teams: an agent architecture for optimization and decision-support, in: Intelligent Agents V: Proc. 5th Inter. Workshop Agent Theories, Architectures, and Languages (ATAL), 1999, pp. 261–276.
[18] Q.H. Nguyen, Y.-S. Ong, M.H. Lim, A probabilistic memetic framework, IEEE Trans. Evol. Comput. 13 (June (3)) (2009) 604–623.
[19] M.A. Muñoz, M. Kirley, S.K. Halgamuge, The algorithm selection problem on the continuous optimization domain, in: C. Moewes, et al. (Eds.), Computational Intelligence in Intelligent Data Analysis, SCI, vol. 445, 2013, pp. 75–89.
[20] K. Smith-Miles, J. van Hemert, Discovering the suitability of optimisation algorithms by learning from evolved instances, Ann. Math. Artif. Intell. 61 (2011) 87–104.
[21] A.S. Fukunaga, Genetic algorithm portfolios, in: Proc. IEEE Congr. on Evolutionary Computation, vol. 2, 2000, pp. 1304–1311.
[22] M. Gagliolo, V. Zhumatiy, J. Schmidhuber, Adaptive online time allocation to search algorithms, Lect. Notes Artif. Intell. 3201 (2004) 134–143.
[23] J.A. Vrugt, B.A. Robinson, Improved evolutionary optimization from genetically adaptive multimethod search, Proc. Natl. Acad. Sci. U. S. A. 104 (3) (2007) 708–711.
[24] J. Grobler, A.P. Engelbrecht, G. Kendall, V.S.S. Yadavalli, Investigating the impact of alternative evolutionary selection strategies on multi-method global optimization, in: Proc. IEEE Congr. on Evolutionary Computation, 2011, pp. 2337–2344.
[25] S.Y. Yuen, X. Zhang, On composing an (evolutionary) algorithm portfolio, in: Proc. Genetic and Evolutionary Computation Conference, 2013, pp. 83–84.
[26] S.Y. Yuen, X. Zhang, On composing an algorithm portfolio, Memet. Comput. 7 (3) (2015) 203–214.
[27] K. Tang, F. Peng, G. Chen, X. Yao, Population-based algorithm portfolios with automated constituent algorithms selection, Inf. Sci. 279 (2014) 94–104.
[28] D. Thierens, An adaptive pursuit strategy for allocating operator probabilities, in: Proc. of the Genetic and Evolutionary Computation Conference, 2005, pp. 1539–1546.
[29] Á. Fialho, M. Schoenauer, M. Sebag, Toward comparison-based adaptive operator selection, in: Proc. of the Genetic and Evolutionary Computation Conference, 2010, pp. 767–774.
[30] P. Auer, N. Cesa-Bianchi, P. Fischer, Finite-time analysis of the multiarmed bandit problem, Mach. Learn. 47 (May–June (2–3)) (2002) 235–256.
[31] M. Gagliolo, J. Schmidhuber, Algorithm portfolio selection as a bandit problem with unbounded losses, Ann. Math. Artif. Intell. 61 (2011) 49–86.
[32] T. Zhang, M. Georgiopoulos, G.C. Anagnostopoulos, Online model racing based on extreme performance, in: Proc. Genetic and Evolutionary Computation Conference, 2014, pp. 1351–1358.
[33] S.Y. Yuen, C.K. Chow, X. Zhang, Which algorithm should I choose at any point of the search: an evolutionary portfolio approach, in: Proc. Genetic and Evolutionary Computation Conference, 2013, pp. 567–574.
[34] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.-P. Chen, A. Auger, S. Tiwari, Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization, Technical Report, Nanyang Technological University, Singapore, May 2005 and KanGAL Report #2005005, IIT Kanpur, India, 2005.

[35] X. Zhang, K.F. Fong, S.Y. Yuen, A Novel artificial bee colony algorithm for HVAC optimization problems, Sci. Technol. Built Environ. (formerly HVAC&R Res.) 19 (6) (2013) 715–731.
[36] K. Deb, An efficient constraint handling method for genetic algorithms, Comput. Methods Appl. Mech. Eng. 186 (2–4) (2000) 311–338.
[37] D. Jones, M. Schonlau, W. Welch, Efficient global optimization of expensive black-box functions, J. Glob. Optim. 13 (4) (1998) 455–492.
[38] G.J. Koehler, Conditions that obviate the no-free-lunch theorems for optimization, INFORMS J. Comput. 19 (2007) 273–279.
[39] A. Tiwari, R. Roy, G. Jared, O. Munaux, Evolutionary-based techniques for real-life optimisation: development and testing, Appl. Soft Comput. 1 (2002) 301–329.
[40] R. Roy, S. Hinduja, R. Teti, Recent advances in engineering design optimisation: challenges and future trends, CIRP Ann. – Man. Technol. 57 (2008) 697–715.