A general cost-benefit-based adaptation framework for Multimeme Algorithms

Wilfried Jakob

Karlsruhe Institute of Technology (KIT), Institute of Applied Computer Science (IAI), P.O. Box 3640, 76021 Karlsruhe, Germany

Phone: +49 7247 82-4663

Fax: +49 7247 82-2602

E-mail: wilfried.jakob@kit.edu

Abstract As Memetic Algorithms (MA) are a crossbreed between local searchers and Evolutionary Algorithms (EA) spreading of computational resources between evolutionary and local search is a key issue for a good performance, if not for success at all. This paper summarises and continues previous work on a general cost-benefit-based adaptation scheme for the choice of local searchers (memes), the frequency of their usage, and their search depth. This scheme eliminates the MA strategy parameters controlling meme usage, but raises new ones for steering the adaptation itself. Their impact is analysed and it will be shown that in the end the number of strategy parameters is decreased significantly as well as their range of meaningful values. In addition to this the number of fitness evaluations is reduced drastically. Both are necessary prerequisites for many practical applications as well as for the acceptance of the method by practitioners.

Although the introduced framework is tailored to EAs producing more than one offspring per mating, it is also suited for those with only one child per pairing. So there are no preconditions to the EA for the described adaptation scheme to be applied.

Keywords: adaptation, Memetic Algorithms, strategy parameter reduction, continuous optimisation, Multimeme Algorithms

1 Introduction

From the beginning of the application of EAs to real-world problems hybridisations with some sort of local searchers (LS) or heuristics were used frequently, see e.g. the recommendations of Davis published in 1991 [12]. One important aim of these hybridisations is the reduction of fitness evaluations. This is of great relevance particularly to applications with runtime-intensive fitness calculations frequently based on simulations. Up to now, this issue has not been affected by the increase of hardware performance, as the demand for more complex simulations is constantly growing with faster computers. In most cases, the merging of local searchers into the framework of EAs is done in the form of

Memetic Algorithms, which integrate local search in the offspring production part of an EA. Another reasonable way, which can be combined with MAs, is the application of LS to a fraction of the initial population, see e.g. [22].

The benefit of LS within an MA can be summarised as follows: firstly, the already mentioned reduction of fitness evaluations required to achieve a certain quality and secondly, the possibility to include domain-specific knowledge into the search process. Additionally, offspring of poor performance, which are located in a region of attraction of a (local) optimum, are likely to be ignored in standard EAs, but can survive, if locally improved and support subsequent search. Furthermore, local search can work as a repair mechanism, if the genetic operators produce infeasible solutions in constraint optimisation and these solutions are penalised by reducing their fitness. The drawback consists in the following design questions, the last two of which mainly control the balance between global and local search or between exploration and exploitation:

1. Which local searcher should be used?
2. Should the LS result be used to update the genotype (Lamarckian evolution) or not (Baldwinian evolution)?
3. How are offspring selected to undergo local search?
4. How often should local improvement be applied or to which fraction of the generated offspring per generation? This is often referred to as *local search frequency*.
5. How long should the local search be run? This results in *local search intensity*.

These questions can be answered in the design phase of an MA resulting in hard-wired solutions. Alternatively, they are left open as new strategy parameters, which then must be adjusted properly. The relevance of local search frequency and intensity has already been described by Hart in 1994 [18] and later by Land [32], who extended the work to the combinatorial optimisation domain. They showed that the answer is problem-dependent. This also holds for the choice of a suited meme, which can be of crucial importance to success, see e.g. Hart [18], Krasnogor [29], Ong and Keane [42], or our own results [22, 23]. The second question - whether to update the chromosome or not - is discussed controversially in literature, see e.g. [14, 55, 44]. When introducing the basic algorithms used for the experimental study, our answer to this question will be given.

In this paper we present a cost-benefit-based adaptation scheme, which controls the parameters resulting from the remaining four of the above five questions. The beneficial effect of the adaptation scheme was reported about in [24, 25], where also the new strategy parameters resulting from the scheme were introduced and discussed. In this work, the adaptation scheme is extended and the effect of the new strategy parameters is investigated empirically in more detail. The main goal is to find out their relevance. In other words, what is their impact on the performance of the algorithm and does it justify manual tuning? Or can a constant value be assigned to at least some, if not all of them? This would significantly ease the burden of cumbersome tuning of strategy parameters and make Memetic Algorithms much more practical to use even for the less experienced engineer.

After discussing related work in Sect. 2, the cost-benefit-based adaptation scheme is described in detail in Sect. 3. Section 4 is devoted to the basic EA used and the two local searchers, which are integrated in the EA to form two simple MAs and an Adaptive Multimeme Algorithm (AMmA). An MA with one meme and without any adaptivity is called simple MA (SMA). Section 5 contains the experimental results, which are based on a real world-problem and six benchmark functions covering different properties. Half of them are applied on two levels of dimensionality to check whether this has in impact or not. Although we do not intend to compete with EAs designed and tuned for optimising continuous functions on a very high level of precision, our AMmA is checked with twelve functions of the CEC'05 benchmark set [51] to give the reader a comparative overview.

2 Related work

In 2004, Hart, Krasnogor, and Smith summarised the situation as follows [19]: "The question of *when* to apply local improvement heuristics, to *which* individuals in EAs population and *how much* computational effort to devote to them remains unanswered, and more research effort is required to gain the understanding and insights that may lead to guidelines for the design of efficient and effective algorithms." Five years later, this still is an open question and theoretical analysis is in an early stage, as Sudholt pointed out in his analysis of a $(\mu + \lambda)$ MA [50]. He showed that even small changes in local search frequency and intensity might have a tremendous impact on the overall performance of the MA.

In the last years, many different variants of Memetic Algorithms were introduced; see e.g. the book by Hart, Krasnogor, and Smith [20]. A good taxonomy and discussion of important design issues can be found in [30]. A step forward was the introduction of adaptivity to adjust parameters dynamically during the run of an MA rather than to set them to constant values [18, 33, 24, 25]. This also includes the choice of the meme to be applied out of a given set [24, 25, 27, 28, 40, 42] or even to construct them out of basic operations [29]. A good overview and a classification of adaptive (multi-)memetic algorithms can be found in [43].

Another line of development is the usage of other EAs other than the traditional ones like for example MAs based on Differential Evolution [38, 40, 41, 39] and the use of the population diversity frequently measured as fitness diversity to steer the adaptation process [38, 36]. Work was continued in [11, 37] to control adaptively meme selection, population size, mutation rate, and strategy parameters of the used memes, which control their search radius. A good summary of this approach can be found in a book chapter by Tironnen and Neri [54].

According to the definition given in [13] and their extensions in [43], adaptation can either be *static*, *adaptive,* or *self-adaptive*. An adaptation is called *static*, if "no form of feedback is used during the evolutionary search" and it is termed *adaptive*, if feedback from the MA search is applied. *Self-adaptivity* employs the adaptive capabilities of evolution by coding the information to be adapted onto the chromosome like for example the used meme. Additionally, adaptation can be *qualitative* or *quantitative* and *local* or *global*. "In *qualitative adaptation*, the exact value of the feedback is of little importance", while *quantitative adaptation* relies on the amount of meme improvements. And finally, *local adaptation* makes no use of historical knowledge, while *global* does [43]. According to this definition, the diversity-based adaptation of an MA based on Differential Evolution mentioned in the last paragraph as well as the cost-benefit-based approach described in the next section can be classified as *quantitative global adaptation*.

In contrast to this more external adaptation mechanism, Krasnogor, Smith and others [27-29, 48] successfully used self-adaptation mainly in the domain of combinatorial optimisation and for meme selection, but recently also for controlling the population size and the genetic operators [49]. A remarkable result

Smith found is that the best meme varies during the course of evolution [49], which supports the idea of dynamically adapting the meme choice during an EA run.

Despite successful applications of self-adaptation there is the drawback that no external measure like fitness diversity or costs can be taken into account. Self-adaptation has proved its suitability when used to control “cost-free” properties like for example the mutation step size of an Evolution Strategy. The choice of different LSs or different search intensities, however, implies different numbers of evaluations (costs) and these costs should be taken into account. An external control mechanism, such as the cost-benefit-based adaptation introduced in this paper, is an appropriate answer. Krasnogor et al. [28] state: “The rationale is to propagate local searchers (i.e. memes) that are associated with fit individuals, as those individuals were probably improved by their respective memes.” Another option is that they were improved by recent evolution. This cannot be judged from the fitness value alone, because the fitness after meme application sums up the fitness coming from evolution and that originating from local search. We think that these are arguments in favour of the investigation of the cost-benefit-based approach or of other dynamic adaptive strategies.

The cost-benefit-based adaptation scheme used by Ong and Keane [42] is close to that used here and it was obviously developed in parallel (cf. [23]). However, Ong and Keane use it for meme selection only and the number of allowed evaluations per LS run is fixed to 100, a comparably low number for some of the nine local searchers involved. Ong’s and Keane’s research on meme selection in the continuous optimisation domain was continued in [43]. As the work presented here focuses more on the intensity and frequency of LS usage, both complement each other.

The approach by Zitzler et al. [57] is based on a fixed time budget. They work in the field of combinatorial optimisation tasks and use parameterised local searchers, where the amount of iterations and, thus, local search intensity can be controlled. They increase the intensity starting with low values according to a given schedule comparable to simulated annealing. The approach was continued and enhanced by Bambha et al. [5], who kept the fixed time frame, but replaced the constant by dynamic schedules, which take the observed increase of solution uality produced by the LS into account. Their detailed investigation demonstrates the superiority of adapting the local search intensity over SMAs (simple MAs based on one meme and without adaptivity). We share the idea of adaptively increasing the precision of local search in the course of evolution. The main difference is the fixed time budget, upon which their algorithm is constructed. Especially for new problems, only rough estimates of the time required for an at least feasible solution can be made. Despite the fact that there are applications where adhering to a limited time is essential, we think that a more general approach which attempts to yield the best result within the shortest possible time, also has its significance.

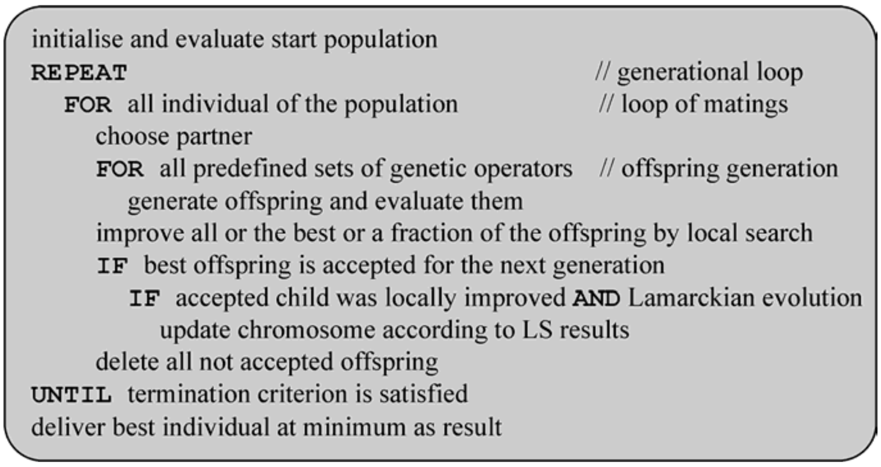
The state of the art of adaptive memetic computing is very well summarised by Meuth et al. [34], where the viewpoint is enlarged to high-order meme-based learning. From a more general perspective (adaptive) memetic algorithms are related with meta- and hyper-heuristics. While meta-heuristics focus on the combination of heuristics complementing each other, hyper-heuristics are aimed at intelligently choosing the right heuristic in a given situation. A good overview is given by Burke et al [10]. As an example of an application of the more basic approach of meta-heuristics motivated by real-world problems the work of Takahara et al. [53] can be mentioned. They tackle the task of waste-minimal allocation of non-convex polygons to a sheet, which arises in industrial fields like textile or sheet metal industry. The broad range of the hyper-heuristic approach is underlined by for example, the work of Bader-El-Den et al. [3], who developed a grammar-based Genetic Programming heuristic framework to evolve constructive heuristics for the timetabling problem.

Another way of introducing adaptivity into evolutionary algorithms is the mutation only genetic algorithm (MOGA) from Szeto and Zhang [52], where the mutation probability is a function of time, fitness, and locus. The approach was extended by re-including the crossover operator based on a hamming distance [31] and was applied successfully to the one-dimensional Ising spin glass problem.

3 Cost-benefit-based adaptation

The basic idea is to steer the adaptation by the costs and the benefit of an LS run. The costs are measured by counting the fitness evaluations required until the LS terminates. To steer search intensity, suited LS must have an external controllable termination parameter like an iteration limit or, even better, a convergence-based termination threshold. Possible additional costs resulting from calculations required by the LS can be neglected, because for real-world problems fitness evaluations, in the majority of cases, consume most of the time of the optimisation procedure. The benefit is measured by the fitness gain as described in the next section.

In the following sections, we will describe how to adaptively control meme selection, local search intensity, and frequency and discuss how offspring are selected for local improvement. Figure 1 shows the pseudocode of the SMA, which serves as the basis for the integration of the adaptation scheme. Within this SMA, it is left open, which part of the offspring is locally improved: all, the best, or some fraction. Furthermore, only the best offspring may be accepted here to substitute its parent as described in Sect. 4.2. Other EAs may use different mechanisms of offspring acceptance, of course.



```
initialise and evaluate start population
REPEAT                                              // generational loop
   FOR all individual of the population             // loop of matings
      choose partner
      FOR all predefined sets of genetic operators  // offspring generation
         generate offspring and evaluate them
      improve all or the best or a fraction of the offspring by local search
      IF best offspring is accepted for the next generation
         IF accepted child was locally improved AND Lamarckian evolution
            update chromosome according to LS results
      delete all not accepted offspring
UNTIL termination criterion is satisfied
deliver best individual at minimum as result
```

Fig. 1 Pseudocode of the simple Memetic Algorithm (SMA)

3.1 Relative fitness gain measurement

As a certain amount of fitness improvement is much easier to achieve in the beginning of a search than in the end, the fitness gain must be measured relatively to the quality already accomplished. For this purpose, a normalised fitness function in the range of 0 and f_{max}, which turns every task into a maximisation problem, is used. f_{max} can be regarded an upper estimation of the fitness[1]. The relative fitness gain *rfg* is defined as the ratio between the observed fitness improvement and the maximum possible one, as Eq. (1) shows.

$$rfg = \begin{cases} \dfrac{f_{LS} - f_{evo}}{f_{\max} - f_{evo}} & \text{if } f_{LS} > f_{evo} \\ 0 & \text{else} \end{cases} \tag{1}$$

f_{LS} is the fitness obtained by the LS and f_{evo} the fitness of the offspring as produced by the last evolutionary step and before it is locally improved. For some LS like those following the logics of simulated annealing, impairments are possible. In this case, *rfg* is set to zero.

3.2 Adaptation of the meme selection probability

To control *n* local searchers LS_1, ..., LS_n, an initially equal selection probability is assigned to every LS. The probabilities of applying the local searchers are adjusted, if either there have been $matings_{max}$ matings in total since the last adjustment or each LS was used at minimum $usage_{LS,min}$ times. The new relation between the memes is calculated as given by Eq. (2).

$$\frac{\sum rfg_{LS_1,i}}{\sum eval_{LS_1,i}} : \frac{\sum rfg_{LS_2,j}}{\sum eval_{LS_2,j}} : \ldots : \frac{\sum rfg_{LS_n,k}}{\sum eval_{LS_n,k}} \tag{2}$$

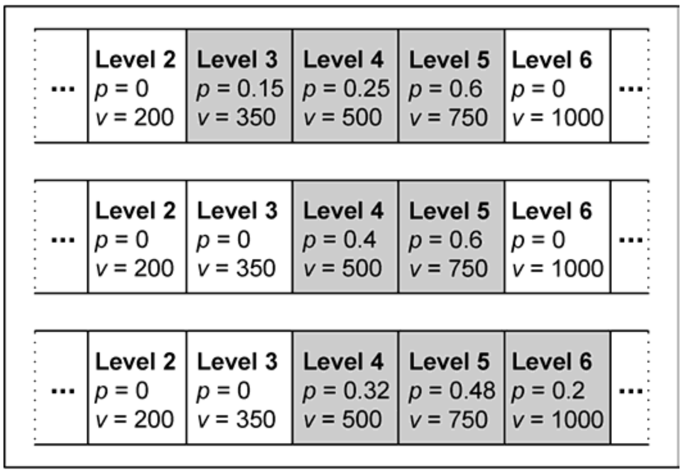
The sums are reset to zero after the adjustment. If the probability for a meme is less than p_{min} for three consecutive alterations, it is ignored from then on. To avoid premature deactivation, the probability is set to p_{min} when it first drops below p_{min}. Despite this precaution, however, erroneous deactivations of a meme were observed. Consequently, a variant of this *simple meme adaptation* was introduced. The *extended meme adaptation* refers to the old distribution by calculating the resulting one as the sum of one third of the old probabilities and two thirds of the

[1] For all EA applications we have done so far in the last 20 years, usage of an upper estimation of the fitness never was a problem. We use this estimation, because all our real-world applications are multi-criteria optimisations, where we used an extended version of the weighted sum. They are from such different areas like collision-free robot path planning, sequence planning, design optimisation, and job shop scheduling with different types of secondary conditions. They comprise continuous, mixed integer, pure integer, and combinatorial optimisation as well as mixtures thereof. Thus, we do not expect that this procedure will limit practical applications.

newly calculated ones according to Eq. (2). Thus, two kinds of *meme selection* will be compared in the experiments, the *simple* and the *extended* one.

3.3 Adaptation of the local search intensity

As already mentioned, the search intensity can be controlled by an iteration limit and, if available, by a termination threshold. To do this, different values must be specified like for example the values of 100, 200, 350, 500, 750, 1000, and so on for the maximum allowed iterations. The definition of these values denoted by v depends on the local searcher in hand and on the experience available. Of the values v, a suited one must be selected before the meme, to which this strategy parameter belongs, is applied. For this purpose, each v has a probability p associated with it, which determines the chance of the associated v to be selected. These pairs of v and p are called *levels* of a strategy parameter P_n. Each P_n has its own set of levels consisting of an appropriate number of pairs. Three consecutive levels are always active[2], i.e. have a probability p greater than zero, and of them, a value v is selected according to the probabilities. They are initialised to meet this requirement as described later. The upper row of Fig. 2 shows an example where the levels with the values 350, 500, and 750 are active.



...
Level 2 $p = 0$ $v = 200$
Level 3 $p = 0.15$ $v = 350$
Level 4 $p = 0.25$ $v = 500$
Level 5 $p = 0.6$ $v = 750$
Level 6 $p = 0$ $v = 1000$
...
Level 2 $p = 0$ $v = 200$
Level 3 $p = 0$ $v = 350$
Level 4 $p = 0.4$ $v = 500$
Level 5 $p = 0.6$ $v = 750$
Level 6 $p = 0$ $v = 1000$
...
Level 2 $p = 0$ $v = 200$
Level 3 $p = 0$ $v = 350$
Level 4 $p = 0.32$ $v = 500$
Level 5 $p = 0.48$ $v = 750$
Level 6 $p = 0.2$ $v = 1000$

Fig. 2 The three phases of one level movement. p denotes the probability of the levels and v the associated strategy parameter values, which are just an illustrating example here. A grey background marks active levels.

[2] Other numbers of levels > 1 are possible too, of course. We found the number of three to be sufficient and well-suited to allow for a movement of the levels of a strategy parameter as described later.

The probabilities of the levels are adjusted in a similar way to the probabilities of the memes. If a level of a strategy parameter P_n is selected to parameterise an LS run, the required evaluations *eval* are counted and the obtained *rfg* is calculated and summed up. New probabilities are calculated, if either all three active levels of that P_n have been used $usage_{L,max}$ in total since the last adjustment or each of them was used at minimum $usage_{L,min}$ times. The new relation among the active levels *L1, L2,* and *L3* of parameter P_n is calculated as shown in Eq. (3).

$$\frac{\sum rfg_{P_n,L1,i}}{\sum eval_{P_n,L1,i}} : \frac{\sum rfg_{P_n,L2,j}}{\sum eval_{P_n,L2,j}} : \frac{\sum rfg_{P_n,L3,k}}{\sum eval_{P_n,L3,k}} \qquad (3)$$

The adjusted probabilities may have the values as shown in the upper row of Fig. 2, where higher levels have a larger probability, so that a tendency to larger values *v* can be assumed. To increase the range of active values *v*, the three active levels must slide to the right and to decrease it, they must be shifted to the left. This movement of active levels is done as follows: if the probability of the lowest or highest active level exceeds a threshold value of 0.5, the next lower or higher level, respectively, is added. To retain the number of three active levels, the level at the opposite end is dropped and its likeliness is added to its neighbour. The new level receives a low likeliness (here 0.2) and the probabilities of the other two levels are rescaled correspondingly. This causes a move of the three consecutive levels along the scale of possible ones according to their performance determined by the achieved fitness gain and the required evaluations. To ensure mobility in both directions, none of the three active levels may have a probability below a minimum, which we choose to be 0.1. If a lower value is calculated, the likeliness is set to 0.1 and the other probabilities are lowered accordingly. This preserves the chance of being selected for every level and thus, the possibility of yielding better results than before. Fig. 3 shows the pseudocode for the level adjustment described in the case of a movement to the right and Fig. 2 gives an example of a movement of the three active levels in the same direction. The shown movement is necessary, because *p* of level 5 has become too large (first row). Level 3 is deactivated (*p*=0) and level 4 inherits its likeliness totalling now *p*=0.4 (second row). Finally, the new level 6 receives 20% of the probabilities of the two others as shown in the third row.

```
IF enough usages of all or each active level of a strategy parameter
   calculate new propabilities p[level], p[level+1], and p[level+2] for the three active levels
   IF p[level+2] > 0.5 AND level < (levelMax - 2)
      p[level+1] = p[level+1] + p[level] // add p of lowest level to its neighbour
      p[level] = 0                       // deactivation of lowest level
      p[level+3] = 0.2                   // activation of new level
      p[level+1] = p[level+1] * 0.8      // rescale the probabilities
      p[level+2] = p[level+2] * 0.8      // of the other two levels
      level = level + 1                  // active levels have moved rightwards
   IF any probability of the three active levels is below 0.1
      set it to 0.1 and rescale the others
```

Fig. 3 Pseudocode of level movement. For the sake of better readability, the code for a movement to the right is given here only. “level” is the index of the first active level L1 and “levelMax” is the number of levels. “p[idx]” denotes the probability *p* of the level of index “idx”.

Note that every level movement must be accompanied by a relative fitness gain, which is proportional to the increased effort to survive the subsequent level adjustment at least. In the above example, a greater amount of allowed iterations must be compensated by an increase in relative fitness gain, which is at least proportional to the enlarged effort. Otherwise, it is too costly and will be reduced in the subsequent adaptations.

If the maximum or minimum of levels is reached, no further movement, and by that no better adaptation, is possible. In this case, a warning is given to the user indicating that the expectations about the meaningful range of values for that strategy parameter are wrong and that the range should be changed accordingly. Thus, it is wise to allow a somewhat larger range of values than those assumed meaningful. As the range of used levels is given together with the probabilities *p* of the final levels at the end of each run, the user receives feedback about his estimations.

Fig. 4 shows how adaptation of meme selection and search intensity fits into the pseudocode of the SMA given in Fig. 1. The line “improve … offspring by local search” is replaced in Fig. 4 by the code marked with a dark grey background. In the first line, the meme and its strategy parameters are chosen according to the actual probabilities. Next, just the best child is locally improved which means that the local search frequency here is simply set to the improvement of the best offspring per mating. We will come back to this in the next section. The subsequent code is for recording the effort and benefit and the adjustment of the meme and level probabilities as described.

```
initialise and evaluate start population
REPEAT                                                   // generational loop
  FOR all individual of the population                   // loop of matings
    choose partner
    FOR all predefined sets of genetic operators         // offspring generation
      generate offspring and evaluate them
    choose meme and its strategy parameters
    improve best offspring by local search
    add effort and relative fitness gain to their sums
    IF enough matings OR enough meme runs of each meme
      adjust meme selection probabilities
      reset sums
    FOR all strategy parameters
      IF enough usages of all or each active level of that strategy parameter
        adjust levels of that strategy parameter
        perform level movement if necessary
        reset sums
    IF best offspring is accepted for the next generation
      IF accepted child was locally improved AND Lamarckian evolution
        update chromosome according to LS results
    delete all not accepted offspring
UNTIL termination criterion is satisfied
deliver best individual at minimum as result
```

Fig. 4 Pseudocode of the Adaptive Multimeme Algorithm (AMmA) with adaptive meme selection and adaptive search intensity. The search frequency is fixed here, as exactly the best child of every mating undergoes local improvement. A dark grey background marks the adaptive part.

As in most cases, local search can start coarsely and become more precise during the course of evolution. Consequently, level probabilities may start with a comparably high value for the lowest level and with decreasing ones for the next two higher levels. For example, good initial values for the level probabilities of the iteration limit of an LS can be 0.5 for the first, 0.3 for the second, and 0.2 for the third level. If a convergence-based threshold is available in addition, it must be borne in mind that both parameters depend on each other. As it is harder and therefore takes longer to adapt two dependent parameters more or less synchronously, it may be wise to use the described initial distribution for one of them only and assign equal initial probabilities to the other.

3.4 Adaptation of the local search frequency

For EAs that create more than one offspring per mating, such as the one used here, a choice can be made between locally optimising the best offspring only (called *best-improvement*) or a fraction of up to all offspring of a mating (called *adaptive all-improvement*). In the latter case, the magnitude of the fraction is controlled adaptively in the following way: the best offspring always undergoes

LS improvement and for its siblings the chance of being treated by the meme is determined by the new strategy parameter p_{all} (p_all in Fig. 5). p_{all} is adjusted using level adaptation as described in the last section, but with the following peculiarities. After having processed all selected offspring, f_{LS} is the fitness of the best locally improved child and f_{evo} that of the best offspring from pure evolution as before. The effort is, of course, the sum of the evaluations required by all meme runs done for all selected offspring of the mating. Fig. 5 shows the pseudocode of the resulting AMmA, which adaptively controls meme selection, search intensity, and frequency. The code for adjusting meme probabilities or levels, which is displayed in Fig. 3 in detail in the then-parts of the if-clauses starting with “IF enough …”, is summarised here as “perform … adjustments … if necessary” for better readability. Note that for *adaptive all-improvement*, all offspring of a mating are treated by the same meme, but with individually selected intensity. The meme must remain the same, because each meme has its own parameter p_{all}, so that the LS frequency is adapted separately for each LS.

```
initialise and evaluate start population
REPEAT                                              // generational loop
  FOR all individual of the population              // loop of matings
    choose partner
    FOR all predefined sets of genetic operators    // offspring generation
      generate offspring and evaluate them
    IF best_improvement                             // best-improvement
      choose meme and its strategy parameters
      improve best offspring by local search
      add effort and relative fitness gain to their sums
      perform adjustments if necessary
    ELSE                                            // adaptive all-improvement
      choose meme and p_all
      FOR all offspring
        IF offspring is best from evolution OR with probability p_all
          choose strategy parameters of selected meme
          improve offspring by local search
          add effort and relative fitness gain to their sums for
            meme selection and meme strategy parameters
          add effort to the sum for p_all of selected meme
          perform level adjustments for strategy parameters if necessary
      calculate and sum up relative fitness gain for p_all of selected meme
      perform adjustments for meme selection and p_all if necessary
    IF best offspring is accepted for the next generation
      IF accepted child was locally improved AND Lamarckian evolution
        update chromosome according to LS results
    delete all not accepted offspring
UNTIL termination criterion is satisfied
deliver best individual at minimum as result
```

Fig. 5 Pseudocode of the Adaptive Multimeme Algorithm (AMmA) with adaptation of meme selection, search intensity, and frequency. The adaptive part marked by a dark grey background includes both alternatives for the selection of the offspring to be treated by the LS: *best-improvement* and *adaptive all-improvement*.

For EAs with one child per mating, a slightly modified version of the *adaptive all-improvement* is suggested. In this case, the individuals to be locally improved are selected from the set of offspring of an entire generation. As this is usually a much greater group than the offspring resulting from a single mating, it is meaningful to decide about the meme for each selected individual separately. Thus, the LS frequency, which is controlled by p_{all}, cannot be associated to the memes any longer. The meme-independent p_{all} will be readjusted now after every generation according to the overall effort and benefit. It might also be meaningful to substitute the random selection of the siblings of the best offspring by a ranked selection to give better offspring a greater chance of being selected for local improvement.

4 Basic algorithms

For the experiments we used the Evolutionary Algorithm GLEAM (General Learning Evolutionary Algorithm and Method) [6, 8] and two local searchers that will be described briefly in the next sections.

4.1 Evolutionary Algorithm GLEAM

GLEAM is an EA of its own, which combines attributes from Evolution Strategy (ES) and Genetic Algorithms (GA) with the concept of abstract data types for the easy formulation of application-oriented genes and chromosomes [6, 8]. The coding is based on chromosomes consisting of problem-configurable gene types. The definition of a gene type constitutes its set of real, integer or Boolean parameters together with their ranges of values. For the test functions, the genes are simply real values and the chromosomes are comparable to those from a real-coded GA. In case of the real-world problem serving as a test case for the experiments the coding is described in Sect. 5.1.

GLEAM uses ranking-based mate selection and elitist offspring acceptance, which is outlined in the next section. In ES notation, it is a $(\mu + \lambda)$ strategy, where λ is usually about 5 to 8 times larger than μ.

4.2 Diffusion model, Lamarckian evolution, and the danger of premature convergence

Instead of the frequently used panmictic population, which is characterised by the ability of each individual to choose any other individual as a partner, the population of GLEAM is based on a diffusion model of overlapping local neighbourhoods [15]. The individuals are located on a ring implying a *geographical* neighbourhood, for example the two individuals to the right and to the left of individual "X" in Fig. 6. Together with "X", they form the so-called *deme* of "X". It should be noted that this kind of neighbourhood must not be confused with those defined by the fitness landscape or the genotypic properties of individuals. Mates can be selected from the deme only and offspring acceptance also is based on the quality of the deme members only: an offspring must have a better fitness than the weakest deme member to replace its parent, provided its parent is not the local best. If it is, the offspring must be better than its parent to ensure the elitist nature of this acceptance rule.

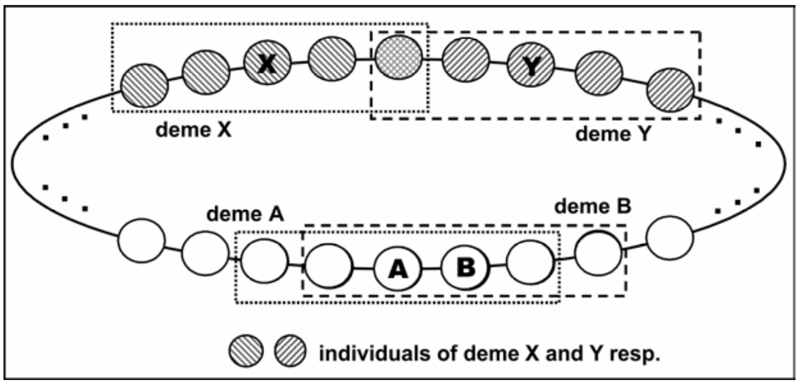


Fig. 6 Diffusion model based on overlapping neighbourhoods, called demes. The demes of the two individuals "X" and "Y" show minimal overlapping, while "A" and "B" overlap maximally.

Since demes overlap, as shown in Fig. 6, genotypic information can spread. As this spreading is much slower as with panmictic populations, niches of more or less similar individuals can emerge, develop, disperse, meet each other, and compete. Thus, genotypic diversity is preserved over a longer period. Furthermore, neighbourhood models like this one cause an adaptive balance between exploration and exploitation [15]. When applied to standard ES, the model produced better results than the original panmictic ES [16, 17].

The diffusion model is the reason why we found Lamarckian evolution superior to Baldwinian [22, 23] and therefore the chromosomes are updated in our AMmA.

Thus, the two if-clauses at the end of the given pseudocodes for the SMA (Fig. 1) and AMmA (Fig. 4 and 5) can be summarised to "IF best offspring is accepted AND was locally improved". The common argument against Lamarckian evolution is the observed tendency to premature convergence especially when panmictic populations are used [14, 55, 44]. As the diffusion model averts this danger, Lamarckian evolution can be used to preserve the results found by the memes within the individual.

4.3 Local search procedures

In many real-world applications, local searchers incorporate some sort of domain knowledge or heuristics. As this turns the generally applicable EA into a domain-specific MA, we decided to use two problem-independent procedures for the experiments. To preserve the general applicability of the resulting MA, they are derivative-free and able to handle restrictions. Due to the lack of space and as Schwefel [47] gives a detailed description of the two selected local procedures together with experimental results, they are explained here briefly only.

4.3.1 Rosenbrock algorithm

Rosenbrock [46] modified the well-known coordinate strategy by rotating the coordinate system so that it points in the direction that appears to be most favourable. For this purpose, the experience of failures and successes is gathered in the course of the iterations. The remaining directions are fixed to be normal to the first one and mutually orthogonal. A direct search is done parallel to the axes of the rotated coordinate system. The procedure stops when the rate of changes of the main search direction decreases below a certain value and when the distances covered become too small. These two quantities are summarised such that the algorithm can be controlled by one convergence-based strategy parameter, called th_R. The implementation on hand uses normalised object parameters in the range between zero and one, thus allowing for the definition of problem-independent threshold values for the termination in addition to the iteration limit.

4.3.2 Complex procedure

The Complex method of Box [9] is based on the SIMPLEX strategy of Nelder and Mead [35], which was improved to handle constraints (COnstrained siMPLEX).

The idea is to use a polyhedron of n+1 to 2n vertices (n is the number of dimensions), the worst vertex of which is reflected at the midpoint of the remaining vertices. The resulting line is lengthened by a factor of $\alpha > 1$, thus expanding the polyhedron. If this leads to an improvement, the worst vertex is replaced by the new one. Otherwise, the polyhedron is contracted. The algorithm stops when no improvement is achieved in five consecutive iterations. As the stopping criterion of this procedure is left fixed in the implementation used, there is only one strategy parameter: the amount of allowed iterations. Schwefel [47] reports about Box's investigations of the effect of the number of vertices and the value of α . He found that they have a no significant effect on the efficiency of his procedure and recommended to set α to 1.3 and to use a low number of vertices, if $n > 5$.

The Complex procedure is abbreviated by *C* and the Rosenbrock algorithm by *R* in the following sections.

4.4 Strategy parameters

The AMmA introduced controls the selection of the two memes and five strategy parameters, which affect the intensity of local search (th_R, $limit_R$, and $limit_C$) and the frequency of meme application ($p_{all,R}$ and $p_{all,C}$), as shown in Table 1. The number of levels and the specific values for th_R, $limit_R$, and $limit_C$ are based on experience gained from the use of these procedures. As the scheduling task works with integer parameters (cf. Sect. 5.1.2), it requires th_R values differing from those of the test functions. For the experiments we used values between 0.8 and 0.01 as shown in Table 1.

For the two p_{all} probabilities, earlier experiments showed that six levels are sufficient. They allow for a minimum probability of 0.06 for a sibling to be improved and of 0.94 at the maximum (level probabilities of 0.8, 0.1, and 0.1 for the three lowest levels or 0.1, 0.1, and 0.8 for the highest). Both are close to the improvement of the best offspring only or of all, so that the complete range is nearly covered. As all strategy parameters are adapted independently, all combinations thereof are covered.

Table 1 Adaptively controlled strategy parameters of the multimeme algorithm

Strategy parameter	Values used for the experiments									
level number	1	2	3	4	5	6	7	8	9	10
th_R (test functions)	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}	
th_R (scheduling task)	0.8	0.7	0.55	0.4	0.25	0.1	0.07	0.04	0.01	
$limit_R$	100	200	350	500	750	1000	1250	1500	1750	2000
$limit_C$	50	100	150	200	300	400	500	650	800	1000
$p_{all,R}$, $p_{all,C}$	0	0.2	0.4	0.8	1.0					

Adaptation starts with the first three levels set active using equal level probabilities *p* for the th_R and the p_{all} strategy parameters. In case of the iteration limits, the probabilities *p* are initialised with 0.5, 0.3, and 0.2 to start the search more coarsely, cf. considerations at the end of Sect 3.3.

As described in Sect. 3, the adaptive scheme introduces some new strategy parameters. The $usage_{LS,min}$, $matings_{max}$, $usage_{L,min}$, and $usage_{L,max}$ parameters described in sections 3.2 and 3.3 control the *adaptation speed* and are combined for *fast*, *medium*, and *slow* adaptation as shown in Table 2.

Table 2 Settings for the new strategy parameters controlling the *adaptation speed*

Adaptation speed	Meme selection		Parameter adaptation	
	$usage_{LS,\ min}$	$matings_{max}$	$usage_{\ L,min}$	$usage_{L,max}$
fast	3	15	3	12
medium	5	20	4	15
slow	8	30	7	25

For the experiments, the lower limit for the probability of a meme p_{min} is set to 0.1.

The experiments reported in [24] gave rise to separate adaptations for different fitness ranges, i.e. different phases of the evolution. Analysis of these experiments showed that the amount of these ranges is of minor influence. Consequently, the effects of *common* and *separate adaptation* using three fitness ranges (0-40%, 40-70%, and 70-100% of f_{max}) are compared in the experiments reported here. Together with the strategy parameters already introduced in sections 3.2 and 3.4, four new strategy parameters result, as shown in Table 3. For the SMA, the effect of improving all or just the best offspring was investigated earlier [23, 24, 25] and the results are summarised here. We use the term *static all-improvement* to distinguish it from its adaptive counterpart.

Table 3 Strategy parameters of the SMA and the AMMA. The new parameters controlling the adaptation are in italic.

Strategy parameter	Relevance and treatment	
	SMA	AMmA
population size μ	manual	manual
Lamarckian or Baldwinian evolution (cf. Sect. 4.2)	manual	Lamarckian evolution
best- or static all-improvement (cf. Sect. 3 and Fig.1)	manual	-
best- or *adaptive all-improvement* (cf. Sect. 3.4)	-	*manual*
LS selection (cf. Sect. 3.2)	manual	adaptive
LS iteration limit (cf. Sect. 3.3)	5000	adaptive
Rosenbrock termination threshold th_R (cf. Sect. 4.3.1)	manual	adaptive
probability p_{all} of the *adaptive all-impr.* (cf. Sect. 3.4)	-	adaptive
simple or *extended meme adaptation* (cf. Sect. 3.2)	-	*manual*
common or *separate adaptation* (cf. Sect. 4.4)	-	*manual*
adaptation speed (cf. Sect. 3.2 and 3.3)	-	*manual*

Looking at Table 3, we must state that we have substituted one set of strategy parameters by another one up to now. Thus, the crucial question is what is their impact? Do we have to adjust them properly according to the application or do they have a minor influence only and can be set to a suitable value? Another question is that of the range of meaningful values of the remaining strategy parameters. For example, what is a useful scale for the population size?

5 Experimental results

After an introduction of the test cases, the results of the basic algorithms and the two SMAs are presented, as they serve as a basis of the subsequent comparisons. Thereafter, the impact of the strategy parameters of the adaptation is investigated, resulting in a recommendation for their setting or further treatment. Based on this, the general effect of the adaptation scheme proposed is studied by comparing the best jobs of SMA, best AMmA and the two recommended AMmA variants. Finally, our AMmA is assessed with a subset of the CEC’05 benchmark set.

5.1 Test cases

Appropriate test cases must be representative of real-world applications, their calculation must be comparatively fast for statistical investigations, and the exact or an acceptable solution must be known.

5.1.1 Benchmark test functions

Table 4 shows characteristic properties of six commonly used test functions taken from the GENEsYs collection [4]. Their definitions are included in the appendix. Although Shekel's foxholes and the Rastrigin function are known to be hard for ES and local searchers, rotated versions are employed to avoid regularities in parallel to the coordinate axes. This makes them harder, so that there is more room for improvements by added local search, see [23]. The fractal function with its unknown optimum is really hard, while the Griewank function was added due to its epistatic properties. The test function of Fletcher & Powel is of considerable complexity. Apart from these multi-modal functions, the sphere function was added to check the approach for the case of unimodality, although most applications are of multimodal nature. The sphere function is known to be easy for ES, but hard for traditional GAs. Three of the five multi-modal functions were used at two different dimensionalities to see, whether and how more dimensions slow down the search efficiency. This was not done for Shekel's Foxholes, as it is defined two-dimensionally, and for the fractal function to limit the effort. Yang and Kao reported that the Griewank function becomes easier when it has more than 15 dimensions [56]. Following that, we used this value for the higher dimension instead of 20 as with the others. The target values and, in case of the sphere function, the range as well were chosen such that the basic EA GLEAM could just solve them. The motivation was to have a large scale for improvements by the different MAs.

Table 4 Important properties of the test cases used. The function numbers f*i* of [4] are given with the references. The last row contains the real-world problem described subsequently. The target value of Shekel's Foxholes is the exact minimum. Abbreviations: f: function, r: real, i: integer, uni: unimodal, multi: multi-modal, Impl. restr.: implicit restrictions.

Test Case	Para-meter	Modality	Impl. restr.	Range	Target value
Schwefel's sphere [4, f1]	30 r	uni	no	$[-5*10^6, 5*10^6]$	10^{-2}
Shekel's foxholes [4, f5]	2 r	multi	no	[-500, 500]	0.998004
Gen. Rastrigin f. [4, f7]	5/20 r	very multi	no	[-5.12, 5.12]	10^{-4}
Fletcher & Powell [4, f16]	5/20 r	multi	no	[-3.14, 3.14]	$10^{-5}/10^{-3}$
Fractal function [4, f13]	20 r	very multi	no	[-5, 5]	-0.05
Griewank f. [4, f22]	5/15 r	very multi	no	[-600, 600]	10^{-6}
Scheduling [7]	87 i	multi	yes		

The common differentiation between uni- and multi-modal is of little help in so far, as it makes no difference between a few suboptima and a large quantity of them. Thus, we added "very" to some multi-modal functions in Table 4, although this is not based on a precise definition.

5.1.2 Scheduling and resource optimisation task

The scheduling and resource optimisation task is taken from chemical industry [7] and deals with 87 batches with varying numbers of workers being required during the different phases of each batch. Figure 7 shows a diagram of varying manpower demand in the white chart. The batch starts with four workers, which are reduced to two and then to one and so on. The objective of scheduling these batches means the best reduction of the overall production time (makespan) as well as the maximum number of workers per shift (human resource). Restrictions like due dates of batches, necessary pre-products from other batches, and the availability of shared equipment also must be observed.

Figure 7 shows an example of the summed up work force demand of two batches. The figure underlines that appropriate starting times of the batches can significantly reduce or increase the peak of workers required within a shift. Thus, it is not sufficient to generate an appropriate sequence of processing the batches. Instead of this common solution to scheduling, suitable starting times must be determined and the relevance of the sequence of genes representing the batches is reduced, as it is used for solving allocation conflicts only. Therefore, a gene consists of its batch identifier and the starting time coded as an integer.

Evolutionary search mainly concerns these times and the combinatorial aspect is limited.

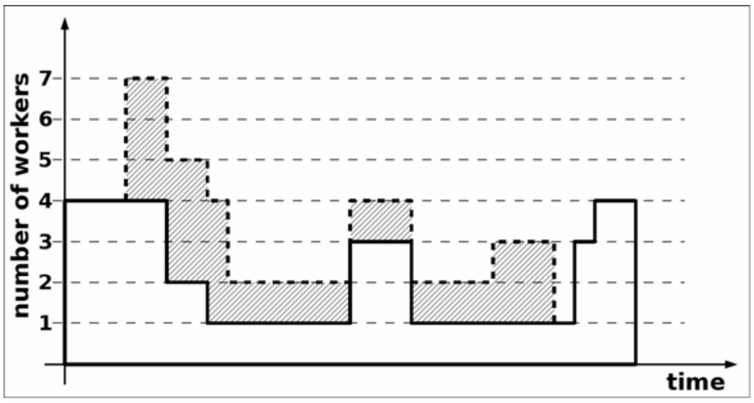


Fig. 7 Cumulated manpower demand of two parallel executed batches. The second batch marked by hachures starts with three workers, continues with one, and ends with two. If it is started later, a peak of five workers instead of seven is achieved.

The optimum is unknown in this case. For the experiments, a reduction from 12 to 9 workers per shift and to 70% of the original production time of 1680 hours for 87 batches given in a manual schedule was regarded sufficient. This is close to the best solution reported in [8].

5.2 Results for the basic algorithms and the two SMAs

A basic algorithm or a hybridisation together with a setting of its strategy parameters is called a *job*. Jobs run until they either reach the target values given in Table 4 or their population converges and the amount of evaluations is recorded. Convergence is assumed, if no offspring is accepted for 500 subsequent generations or no local improvement of its deme has occurred for 1000 subsequent generations (cf. Sect. 4.2). The comparisons are based on the average number of evaluations required from one hundred runs per job. To obtain an impression of the robustness of a job and for a better comparison, the 95% confidence intervals are given as well. Where necessary, t-tests (95% confidence again) are used to distinguish between significant and stochastic differences. With the exception of the two local searchers, only jobs are taken into account, the runs of which are all successful, i.e. reach the target values of Table 4 or the solution quality given in section 5.1.2 in case of the real-world problem.

Table 5 summarises the results for the three basic algorithms[3]. The large spread of the population size demonstrates how difficult it is to find suited ones for new applications. The sphere function is the only one, which can be solved reliably by the Rosenbrock algorithm, although only with an unusually low value for th_R. On the other hand, it is not surprising that a good local searcher can find the optimum of a unimodal test function.

Table 5 Best jobs of the basic algorithms. A grey background indicates the columns of the two strategy parameters, the population size μ, and the threshold of the Rosenbrock algorithm th_R. There is no success for GLEAM in the 20-dimensional cases of the Rastrigin and the Fletcher & Powell functions despite extremely large populations and numbers of evaluations. If the Rosenbrock algorithm fails completely, the table shows no data, as the amount of evaluations increases with decreasing th_R values, but all yield no success. Abbreviations: Conf.I.: 95% confidence interval, Succ.: success rate in per cent, Eval.: evaluations as the mean of 100 runs

Test case (dimensions)	GLEAM			Rosenbrock alg.			Complex alg.	
	μ	Evaluations	Conf.I.	th_R	Succ.	Eval.	Succ	Eval.
Sphere (30)	120	37,163,091	974,707	10^{-8}	100	4,706	0	5,000
Foxholes (2)	300	108,435	8,056	10^{-4}	5	133	0	95
Rastrigin (5)	11,200	3,518,702	333,311		0		0	5000
Rastrigin (20)					0		0	5000
Fletcher (5)	600	483,566	191,915	10^{-8}	22	5,000	26	658
Fletcher (20)					0		0	4,520
Fractal (20)	20	195,129	20,491		0		0	781
Griewank (5)	120	2,275,903	156,651		0		0	200
Griewank (15)	500	61,685,632	2,075,356	10^{-9}	23	3,364	0	3,616
Scheduling	1,800	5,376,334	330,108		0		0	473

Table 6 gives the results for the two SMAs[3]. An important outcome is that the wide range of best population sizes from 20 to 11,200 is narrowed down to 5 to 200 by the best SMAs. Furthermore, the choices between the memes, *best-* and *static all-improvement*, and of a feasible value for th_R are problem-dependent.

[3] The results presented here differ from those reported in [19-21] in three cases. To better show the effects of adaptation, the sphere function is parameterised differently, as described in Sect. 5.1.1. Secondly, a different GLEAM job is used for Shekel's foxholes as a reference, because it has a much better confidence interval and is nearly as good as the job used so far. And thirdly, all MA and AMmA runs for the scheduling task were repeated, because an error was found which affected the memetic part of this application.

Table 6 Best jobs of both SMAs. Confidence intervals are given for jobs better than GLEAM only. A grey background indicates the strategy parameters and the best jobs are marked by boldface. Abbreviations not contained in Table 5: b/a: *best-* or *static all-improvement*

Test case (dimensions)	Rosenbrock-SMA					Complex-SMA			
	μ	th_R	b/a	Eval.	Conf.I.	μ	b/a	Eval.	Conf.I.
Sphere (30)	20	10^{-6}	best	**5,453**	395				
Foxholes (2)	30	10^{-2}	all	10,710	1,751	5	all	**9,360**	1,426
Rastrigin (5)	70	10^{-2}	all	**315,715**	46,365	150	all	3,882,513	
Rastrigin (20)	90	10^{-4}	all	**14,128,935**	812,833				
Fletcher (5)	10	10^{-4}	best	13,535	1,586	5	best	**4,684**	812
Fletcher (20)	30	10^{-7}	all	**1,596,751**	120,690				
Fractal (20)	5	10^{-2}	best	**30,626**	3,353	10	best	1,065,986	
Griewank (5)	200	10^{-2}	all	**1,014,112**	274,268	200	best	2,900,064	
Griewank (15)	70	10^{-6}	best	**16,744**	3,249				
Scheduling	10	0.6	all	**483,579**	36,408				

In all cases, the best SMA achieved an improvement by several factors compared to the basic EA GLEAM. Table 7 compares the best jobs from GLEAM to the better of the two SMAs. It also shows the associated strategy parameters, including the iteration limit, which was omitted in Table 6 due to the lack of space. It must be stressed that these mostly impressive improvements could be achieved only by cumbersome and time-consuming manual tuning of the strategy parameters!

Table 7 Improvements of the best SMA jobs compared to the best GLEAM jobs together with all associated strategy parameters. Abbreviations not contained in Table 5: R: Rosenbrock algorithm, C: Complex procedure, Iter.: iteration limit, b/a: *best-* or *static all-improvement*, Impr.: improvement factor.

Test case (dimensions)	Meme	μ	Iter.	th_R	b/a	Impr. / remark
Sphere (30)	R	20	2,000	10^{-6}	best	6,815
Foxholes (2)	C	5	2,000		all	11.6
Rastrigin (5)	R	70	500	10^{-2}	all	11.1
Rastrigin (20)	R	90	2,000	10^{-4}	all	- only SMA success
Fletcher (5)	C	5	2,000		best	103.2
Fletcher (20)	R	30	2,000	10^{-7}	all	- only SMA success
Fractal (20)	R	5	2,000	10^{-2}	best	6.4
Griewank (5)	R	200	2,000	10^{-2}	all	2.2
Griewank (15)	R	70	2,000	10^{-6}	best	3,249
Scheduling	R	10	1,000	0.6	all	11.1

5.3 Impact of the strategy parameters of the adaptation

Next, we consider the effect of the strategy parameters controlling the adaptation as listed in Table 3. Three of the four parameters can take two values (*best-* or *adaptive all-improvement*, *simple* or *extended meme adaptation*, *common* or *separate adaptation*), while the fourth one, the *adaptation speed*, was set to have one of three values: *small*, *medium*, or *large*. This results in 24 different parameterisations. For each parameter and every value combination of the remaining ones, the best and the worst results expressed in required evaluations are compared and the difference is expressed as a percentage of the best value. This yields, for each strategy parameter, the maximum effect of a "false decision" for every parameterisation of the other parameters. The resulting values (eight for the *adaptation speed* and twelve for every other parameter) are averaged per strategy parameter and test case. These averages are displayed in Fig. 8. As an example, the column of about 200% for *best-/all-improvement* and the sphere function can be interpreted such that the weakest decision will be 200% worse than the best one on the average.

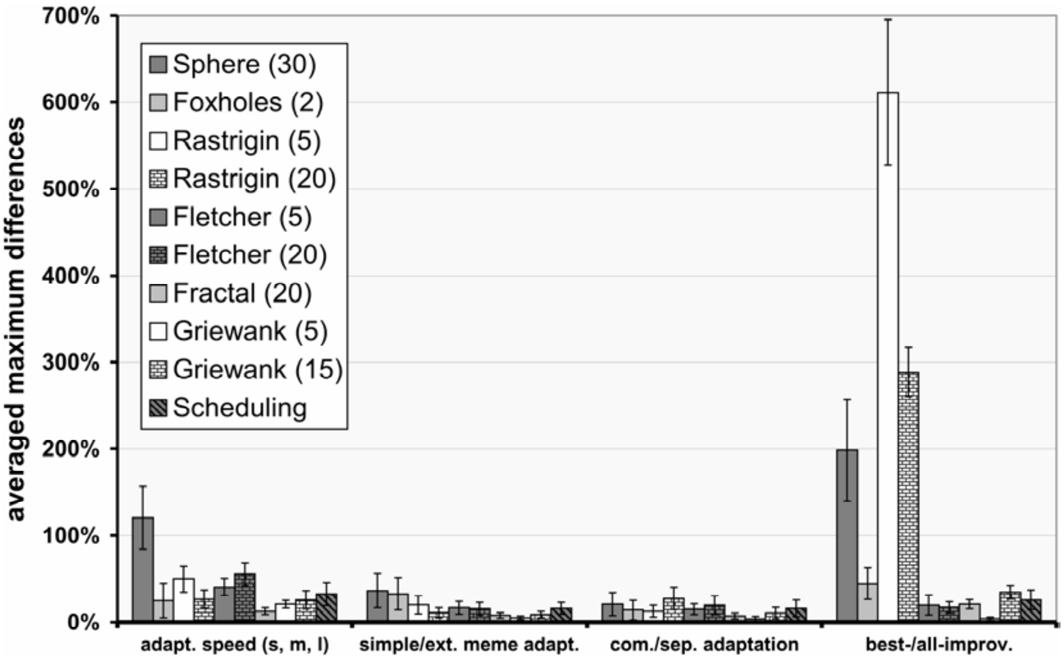


Fig. 8 Impact of the strategy parameters of the adaptation for all test cases. The effect of each strategy parameter is expressed as the worst results of "false selections" averaged over all combinations of the remaining parameters as described in the text. Abbreviations: adapt.: adaptation, s: slow, m: medium, l: large, extend.: extended, com.: common, sep.: separate, improv.: improvement

Figure 8 shows that the choice between *best-* and *adaptive all-improvement* is obviously crucial to at least three test cases, as a false selection can change the outcome in the magnitude of factors. For a better presentation, Fig. 9 contains a partial view of the three other strategy parameters. From Fig. 9, the following conclusion can be drawn: for the range covered by the test cases investigated, the impact of *common* or *separate adaptation* and *simple* or *extended meme adaptation* is of minor impact, while an appropriate choice of the *adaptation speed* may have a relevant influence.

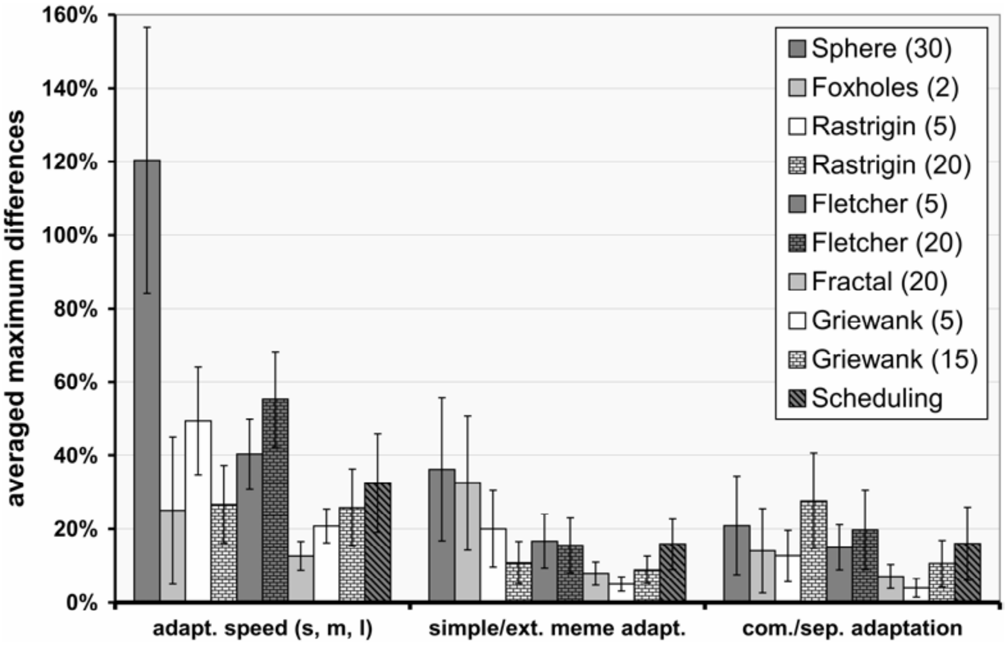


Fig 9 Partial view of Fig. 8 showing the effect of three of the four strategy parameters in greater detail

Thus, the choice between *best-* and *adaptive all-improvement* must be considered as problem-dependent. The next step is an attempt to find a common parameterisation for the remaining three strategy parameters shown in Fig. 9. For a decision on *simple* and *extended meme adaptation*, the fact that a false deactivation of a meme cannot be reversed must be considered. Consequently, a more detailed analysis as it can be presented here shows that with *simple meme adaptation* more unsuccessful runs due to this deactivation occur than with *extended meme adaptation*. Thus, we decide in favour of the extended variant. As the differences between *common* or *separate adaptation* are fairly small, it does not justify the additional implementation effort. Therefore, *common adaptation* is the next choice.

The decision on the remaining parameter is justified as follows: The set of best, second best, and all statistically not differing runs (t-test at 95% confidence) serves as a reference set. Based on the two parameter suggestions given before, the three *adaptation speeds*, and the left-open choice between *best*- and *adaptive all-improvement*, there are six parameterisation candidates. For each candidate and test case, it is checked whether it is included in the reference set. This procedure yields *fast* and *common adaptation* in conjunction with *extended meme adaptation* as the best parameter setting for the test cases investigated here. All, but one test case are within the reference set and only two belong to the set of second best or similar ones. The exception is the scheduling task, where the recommended parameterisation yields a result, which is worse than the best one, but the impairment amounts to 31% only. As the test cases investigated cover a wide range of continuous and integer optimisation problems, the recommendation is a good parameterisation to start with in case of a new optimisation task from this domain.

5.4 Comparison of basic EA, SMA, and AMmA

What can be expected from adaptation: on the one hand, improvement because of better tuned strategy parameters and on the other hand, impairment, because adaptation means learning and learning causes some additional expense. And in fact, both effects can be observed. Table 8 contains the results for the best jobs of the two recommended variants of the AMmA. The range of good population sizes remains the same as with the best SMA jobs, cf. Table 7.

Table 8 Comparison of the results of the best SMA and both AMmA jobs. To facilitate the comparison, confidence intervals (CI) are given as percentage of the mean here. A grey background marks the best AMmA jobs. If the alternative does not differ statistically from the best one, a light grey background is used. Abbreviations not contained in Tables 5 and 6: alt.: alteration factor with respect to the best SMA job, imp.: impairment, see text.

Test case (dimension)	AMmA, *best-improvement*					AMmA, *adaptive all-impr.*				
	μ	Eval.	CI	alt.	imp.	μ	Eval.	CI	alt.	imp.
Sphere (30)	5	54,739	7	0,10	-10.0	20	175,433	9	0,03	-32.2
Foxholes (2)	90	6,553	15	1,43		50	5,566	20	1,68	
Rastrigin (5)	250	2,865,679	17	0,11	-9.1	90	329,938	18	0,96	
Rastrigin (20)	200	21,580,135	12	0,65	-1.5	120	5,375,234	15	2,63	
Fletcher (5)	10	11,451	14	0,41	-2.4	5	14,640	17	0,32	-3.1
Fletcher (20)	200	3,382,648	11	0,47	-2.1	150	4,205,266	8	0,38	-2.6
Fractal (20)	10	19,904	14	1,54		10	24,489	10	1,25	
Griewank (5)	70	203,707	19	4,98		70	212,857	16	4,76	
Griewank (15)	50	51,821	9	0,32	-3.1	50	72,654	7	0,23	-4.3
Scheduling	50	405,846	16	1,19		20	295,926	12	1,63	

As the number of evaluations shows a great spread, it cannot serve for a graphical comparison. Instead, the alteration is calculated and included in the columns “alt.” in Table 8. For a better presentation, impairments are calculated as $-1/alteration$, such that both improvement and impairment have the same magnitude and hence, columns of the same size in Fig. 9. The difference is given by the sign. The columns “imp.” of Table 8 contain these values. The resulting diagram is shown in Fig. 9, which also shows the results for the best AMmA.

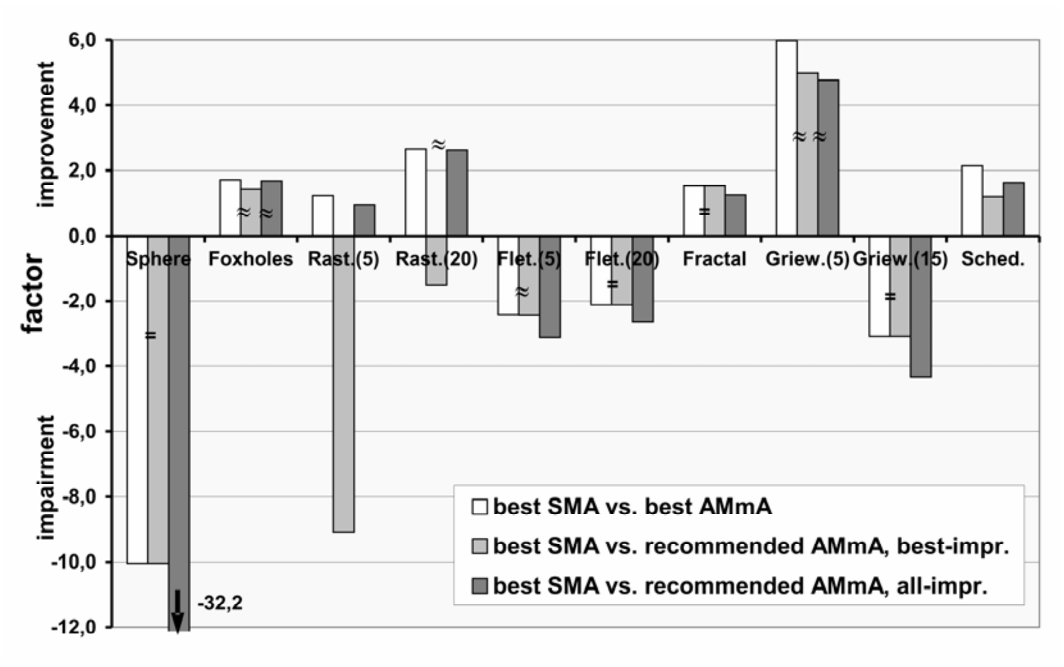


Fig. 9 Comparison of the improvement or impairment factors based on the average efforts of the best SMA jobs. For a better presentation, impairments are shown as negative factors as explained in the text. Statistically not differing jobs are indicated by "≈" and identical ones by "=".

Figure 9 shows that in all cases, but two the best AMmA job and the better of the two recommended ones are either identical or have no statistically significant difference. For both exceptions (Rastrigin (5) and Scheduling tasks), the differences are acceptably small. The figure also shows that for problems with an extreme amount of suboptima like the Rastrigin function the choice between *best-* and *adaptive all-improvement* can be crucial, while in other cases it is of less importance. As this is unknown in advance, both cases should be tested for a new optimisation problem.

To explain the observed impairments, the th_R values of the best Rosenbrock-SMA jobs given in Table 6 are examined. The improved test cases all use th_R values that are within the scope of the initial levels, see Table 1 and Sect. 4.4, while the worsened cases require lower th_R values (higher levels). As the adaptation starts with values of 0.1 to 0.001, a longer phase of adaptation is required to decrease th_R and to increase $limit_R$, which is needed to benefit from lower th_R thresholds. This explanation of the impairments observed is checked for the most drastic case, the sphere function, by starting with medium levels for both strategy parameters. As a result, the impairment factor was reduced to 3.3.

As pointed out before the proposed adaptation scheme can be applied to any set of memes and EA. The only precondition is that the memes provide a parameter for controlling the search intensity. The strategy parameters to be manually tuned are the population size and the choice between *best*- and *adaptive all-improvement*. As Tables 5 and 8 show, good population sizes are comparatively small with respect to the experiences from the basic EA. Their tuning remains crucial in so far, as too small sizes may hinder success (i.e. reaching the required quality) and too large ones are waste of computing power. Table 9 shows the smallest successful and the best population size for both kinds of improvement and test cases investigated here.

Table 9 Smallest successful and best population size for each test case at *best*- and *adaptive all-improvement* (lowest/best). Abbreviations: dim.: dimensions of the test functions

	Sphere	Foxholes	Rastrigin		Fletcher		Fractal	Griewank		Sched.
dim.	30	2	5	20	5	20	20	5	15	
best	5/5	10/90	200/250	200/200	10/10	200/200	5/10	50/70	20/50	50/50
all	20/20	5/50	70/90	90/120	5/5	150/150	5/10	30/70	10/50	10/20

5.5 Comparison with some CEC'05 functions

As pointed out in [6, 8], GLEAM can perform optimisations of real, integer, and mixed-integer parameters, but it was not especially designed for this application domain. As the AMmA presented here is aimed at continuous optimisation due to the nature of the local searchers used, a comparison with a subset[4] of the benchmark functions defined for the CEC'05 conference [51] will be given. However, it must be stressed that our AMmA is neither tuned nor designed for solving continuous optimisation tasks on a very high level of precision. The original goal was just to assess the cost-benefit adaptation algorithm introduced.

In [21] eleven different EAs are evaluated, of which three are selected to compare our results with. These are the very well-performing "restart CMA ES" (G-CMA-ES) [2], the "local restart CMA ES" (L-CMA-ES) [1], which also yields good results, and an "ES with mutation step co-evolution" (CoEVO) [45]. To limit the effort, we use twelve of the set of 25 test functions at 30 dimensions with the given maximum of function evaluations Max_FES $= 3 \cdot 10^5$. The comparison is

[4] We used a subset to limit the effort and because a subset is sufficient to give an impression about the performance, which is attainable by the proposed adaptation scheme applied to a hybrid of basic algorithms, each of which does not compete applied solely.

based on the achieved function error value (FEV), which is the difference between the optimum and the best result of a run. A run is stopped whenever this value drops below 10^{-8} or MaxFES is reached. Table 10 shows the results in a reduced scheme compared to that given in [51] and summarises the results in the rightmost column. The best-performing algorithm clearly is *the restart CMA ES* which performs (almost) best with six functions (f2, f5, f6, f7, f10, f11). As our AMmA shows (almost) best results with three functions (f1, f9, f15) and good ones with two functions (f4, f12), it does not compete too badly.

Table 10 Comparison of the results of twelve test functions from the CEC'05 set for $n = 30$ dimensions. The function numbers are those given in [51]. The AMmA column also indicates whether *best-* or *adaptive all-improvement* was used. A grey background marks the best results. The *Comp.*-column gives the following comparison: the AMmA results are about equal to (=), worse (-), or better (+) than the three other EAs. If the mean obtained from the AMmA is better, but its best result is worse than the EA compared with, a "≈" is given.

Test functions	Statistics	G-CMA-ES	L-CMA-ES	CoEVO	AMmA		Comp.
f1 shifted sphere function (unimodal)	best	3.98e-9	2.98e-9	8.29e-9	2.03e-10	best	= = +
	mean	5.42e-9	5.28e-9	7.97e-1	4.18e-9		
	worst	7.51e-9	7.06e-9	1,18e+1	9.84e-9		
	std. dev.	9.80e-10	9.82e-10	2.49e+0	2.90e-9		
f2 shifted Schwefel's problem 1.2 (unimodal)	best	4.48e-9	4.90e-9	8.80e-9	1.64e-4	best	– – ≈
	mean	6.22e-9	6.93e-9	4.40e-1	6.12e-3		
	worst	8.41e-9	8.77e-9	7.56e+0	2.42e-2		
	std. dev.	8.95e-10	8.27e-10	1.52e+0	4.73e-3		
f3 shifted rotated highly conditioned elliptic function (unimodal)	best	4.07e-9	2.98e-9	2.59e+3	4.25e+4	best	– – –
	mean	5.55e-9	5.18e-9	3.67e+2	2.51e+5		
	worst	8.66e-9	7.57e-9	6.73e+3	8.62e+5		
	std. dev.	1.09e-9	1.03e-9	1.35e+3	2.14e+5		
f4 shifted Schwefel's problem 1.2 with noise (unimodal)	best	6.06e-9	2.31e+1	8.39e+2	9.94e+1	best	≈ + +
	mean	1.11e+4	9.26e+7	4.80e+3	2.85e+2		
	worst	1.31e+5	4.48e+8	1.27e+4	5.02e+2		
	std. dev.	3.02e+4	1.68e+8	3.44e+3	1.18e+2		
f5 Schwefel's problem 2.6 with optimum on bounds (unimodal)	best	7.15e-9	5.23e-9	6.94e+3	1.05e+3	all	– – +
	mean	8.62e-9	8.30e-9	8.34e+3	2.10e+3		
	worst	9.96e-9	9.99e-9	9.94e+3	3.17e+3		
	std. dev.	8.53e-10	1.38e-9	1.04e+3	4.54e+2		
f6 shifted Rosenbock's function (multi-modal)	best	3.27e-9	4.28e-9	7.72e+1	5.17e-4	all	– – +
	mean	5.90e-9	6.31e-9	1.21e+3	1.11e+1		
	worst	9.25e-9	8.44e-9	6.14e+3	2.16e+1		
	std. dev.	1.61e-9	1.14e-9	1.83e+3	7.73e+0		
f7 shifted rotated Griewank's function w/o bounds (multi-modal)	best	1.79e-9	4.46e-9	7.37e-7	2.46e-5	best	– – ≈
	mean	5.31e-9	6.48e-9	1.41e-1	1.44e-4		
	worst	7.81e-9	8.79e-9	9.90e-1	4.17e-4		
	std. dev.	1.41e-9	1.14e-9	2.83e-1	1.18e-4		
f9 shifted Rastrigin's function (multi-modal)	best	4.35e-6	2.43e+2	9.16e+1	2.81e-11	best	+ + +
	mean	9.38e-1	2.91e+2	1.31e+2	2.40e-9		
	worst	4.97e+0	3.66e+2	1.83e+2	7.87e-9		
	std. dev.	1.18e+0	3.54e+1	2.39e+1	2.74e-9		
f10 shifted rotated Rastrigin's function (multi-modal)	best	3.08e-4	4.20e+1	1.69e+2	3.98e+1	all	– + +
	mean	1.65e+0	5.63e+2	2.32e+2	6.27e+1		
	worst	4.59e+0	7.92e+2	2.91e+2	1.24e+2		
	std. dev.	1.35e+0	2.48e+2	3.51e+1	2.31e+1		
f11 shifted rotated Weierstrass function (multi-modal)	best	8.27e-10	6.85e+0	3.41e+1	1.05e+1	all	– – +
	mean	5.48e+0	1.52e+1	3.77e+1	1.56e+1		
	worst	1.14e+1	2.26e+1	4.17e+1	2.03e+1		
	std. dev.	3.13e+0	3.51e+0	1.53e+0	2.75e+0		
f12 Schwefel's problem 2.13 (multi-modal)	best	3.79e-9	1.29e+0	1.35e+4	1.16e+3	all	≈ ≈ +
	mean	4.43e+4	1.32e+4	1.01e+5	1.96e+3		
	worst	1.10e+6	3.50e+4	2.19e+5	4.83e+3		
	std. dev.	2.19e+5	1.15e+4	7.22e+4	1.47e+3		
f15 hybrid composition function (multi-modal)	best	2.00e+2	1.08e+2	2.97e+2	2.54e-9	all	+ + +
	mean	2.08e+2	2.16e+2	4.11e+2	1.24e+1		
	worst	3.00e+2	4.00e+2	4.76e+2	2.00e+2		
	std. dev.	2.75e+1	8.29e+1	4.37e+1	3.97e+1		

6 Conclusions and outlook

An EA-independent cost-benefit-based adaptation procedure for Memetic Algorithms was introduced, which controls both LS selection and the balance between global and local search. The latter is achieved by adapting the intensity of local search and the frequency of its usage. It was shown how this adaptation method fits into the gap previous work in the field left open.

The adaptation scheme presented introduces four new strategy parameters the effect of which was studied using six test functions, three of them at different dimensionalities, and one real-world application, totalling ten test cases. For the range covered by these test problems, the following results were obtained:

- The largest impact on the performance has the choice between *best*- and *adaptive all-improvement* (cf. Sect. 3.4), which controls the local search frequency. An appropriate decision between the two alternatives is problem-dependent and must be left open.
- The choice between *common adaptation* during the entire MA run or a *separated* one for different stages of fitness values (cf. Sect. 4.4) has a minor impact only. To avoid unnecessary implementation effort, *common adaptation* is recommended.
- The same holds for the option between *extended* and *simple meme adaptation* (cf. Sect. 3.2), which controls whether a fraction of the old probability distribution of the memes is used for the calculation of the new one or not. The extended form is chosen, because it yields smoother changes and avoids false and premature deactivation of a meme.
- The *adaptation speed* (cf. Sect. 3.2, 3.3, and 4.4) has an impact for some test cases and it was found that fast adaptation is the best choice.
- The range of appropriate population sizes could be narrowed down from ranges between 20 and 11,200 to ranges between 5 and 200. Thus, significantly fewer sizes must be checked to find a suitable one for a given problem.

As the proposed adaptation method can be applied to any EA other than the one used here for assessment, we hope that it will support the practitioner in applying his favourite EA to new tasks.

Further investigations will take into account more test cases, preferably more real-world applications, and aim at the extension of the method to combinatorial tasks. Namely, job scheduling and resource allocation in the context of grid computing is a subject we are currently working on and for which new memes are required [26].

Appendix

The benchmark functions used are defined as follows (cf. [4], function numbers in brackets).

Schwefel's Sphere (f1):

$$f(\vec{x}) = \sum_{i=1}^{n} x_i^2$$

$$n = 30; \quad -5\cdot 10^6 \le x_i \le 5\cdot 10^6; \quad \min(f(x_i)) = f(0,\cdots,0) = 0$$

Shekel's Foxholes (f5):

$$f(\vec{x}) = \frac{1}{\frac{1}{K} + \sum_{j=1}^{25} \frac{1}{j + (x_1 - a_{1j})^6 + (x_2 - a_{2j})^6}}$$

$$(a_{ij}) = \begin{pmatrix} -32 & -16 & 0 & 16 & 32 & -32 & \cdots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & \cdots & 32 & 32 & 32 \end{pmatrix}$$

$$K = 500$$

$$-500 \le x_i \le 500; \; i = 1,2; \quad \min(f(\vec{x})) \approx f(-31.92, -31.9481) \approx 0.99804$$

For the experiments the function was rotated by 30 degrees.

Generalised Rastrigin function (f7):

$$f(x) = n\cdot A + \sum_{i=1}^{n} (x_i^2 - A\cdot\cos(\omega\cdot x_i))$$

$$n = 5; \quad A = 10; \quad \omega = 2\pi; \quad -5.12 \le x_i \le 5.12; \quad \min(f(\vec{x}) = f(0,...,0) = 0$$

For the experiments the function was rotated by 30 degrees.

Function from Fletcher and Powell (f16):

$$f(\vec{x}) = \sum_{i=1}^{n} \left(A_i - B_i\right)^2$$

$$A_i = \sum_{j=1}^{n} (a_{ij} \sin(\alpha_j) + b_{ij} \cos(\alpha_j))$$

$$B_i = \sum_{j=1}^{n} (a_{ij} \sin(x_j) + b_{ij} \cos(x_j))$$

$$a_{ij}, b_{ij} \in [-100,100]; \quad \alpha_j \in [-\pi, \pi]$$

$$n = 5; \quad -\pi \le x_i \le \pi; \quad \min(f(\vec{x})) = 0$$

$$(a_{ij}) = \begin{pmatrix} -25.5668 & 96.1559 & -15.2661 & -23.247 & 1.15129 \\ -97.5098 & -82.9516 & 50.3364 & -96.552 & 29.4616 \\ 36.3096 & 54.3934 & 76.9586 & 64.3481 & 0.66188 \\ -58.7916 & 30.4881 & -22.9354 & 36.1643 & 62.2798 \\ 66.582 & -18.2818 & 2.30184 & -38.6698 & -80.3784 \end{pmatrix}$$

$$(b_{ij}) = \begin{pmatrix} -38.9994 & -49.1546 & -52.2403 & 66.4195 & -36.6538 \\ -6.49429 & -97.6865 & 39.6734 & -91.4542 & 17.2329 \\ -67.1681 & -42.8325 & -42.9307 & 36.9705 & -18.3463 \\ -53.1271 & -2.74637 & 27.6244 & -72.7537 & 46.1798 \\ -1.47036 & -95.2547 & 78.6214 & -54.3113 & 84.6737 \end{pmatrix}$$

$$(\alpha_j) = (1.98961 \quad -1.39376 \quad -2.03048 \quad -0.621016 \quad 3.12736)$$

Fractal function from Weierstrass and Mandelbrot (f13):

$$f(\vec{x}) = \sum_{i=1}^{n} \left(\frac{C(x_i)}{C(1) \cdot |x_i|^{2-D}} + x_i^2 - 1 \right)$$

$$C(x) = \sum_{j=-\infty}^{\infty} \frac{1 - \cos(b^j x)}{b^{(2-D)j}}$$

$$1 \le D \le 2; \quad b > 1; \quad -5.12 \le x_i \le 5.12$$

$$n = 20; \quad \text{unknown minimum}$$

Griewank function (f22):

$$f(\vec{x}) = 1 + \frac{1}{d} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}} \right)$$

$$n = 10; \quad d = 400; \quad -600.0 \le x_i \le 600.0; \quad \min(f(\vec{x}) = f(0,...,0) = 0$$

References

1. Auger A, Hansen N (2005) Performance evaluation of an advanced local search evolutionary algorithm. In: Conf. Proc IEEE Congr. on Evol. Comp. CEC 2005, pp 1777-1784
2. Auger A, Hansen N (2005) A Restart CMA Evolution Strategy With Increasing Population Size. In: Conf. Proc IEEE Congr. on Evol. Comp. CEC 2005, pp 1769-1776
3. Bader-El-Den M, Poli R, Fatima S (2009) Evolving timetabling heuristics using a grammar-based genetic programming hyper-heuristic framework. J Memetic Comp. 1(3):205-219
4. Bäck T (1992) GENEsYs 1.0. http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/genetic/ga/systems/genesys/0.html Accessed 11 August 2009
5. Bambha NK, Bhattacharyya, SS, Zitzler E, Teich J (2004) Systematic Integration of Parameterized Local Search into Evolutionary Algorithms. IEEE Trans. on Evol. Comput. 8(2):137-155
6. Blume C (1991) GLEAM - A System for Intuitive Learning. In: Schwefel H-P, Männer R (eds): Conf. Proc. PPSN I, LNCS 496, Springer, Berlin, pp 48-54
7. Blume C, Gerbe M (1994) Deutliche Senkung der Produktionskosten durch Optimierung des Ressourceneinsatzes. atp 36(5): 25-29, Oldenbourg Verlag, München (in German)
8. Blume C, Jakob W (2002) GLEAM - An Evolutionary Algorithm for Planning and Control Based on Evolution Strategy. In: Cantú-Paz E (ed): Conf. Proc. GECCO 2002, LBP, pp 31-38
9. Box MJ (1965) A New Method of Constrained Optimization and a Comparison with Other Methods. The Computer Journal, 8, pp 42-52
10. Burke E, Kendall G, Newall J, Hart E, Ross P, Schulenberg S (2003) Hyper-Heuristics: An Emerging Direction in Modern Search Technology. In: Glover F, Kochenberg GA (eds): Handbook of Metaheuristics, Kluwer, pp 457-474
11. Caponio A, Cascella GL, Neri F, Salvatore N, Sumner M (2007) Fast Adaptive Memetic Algorithm for Online and Offline Control Design of PMSM Drives. IEEE Trans. Systems, Man, and Cybernetics, Part B 37(1):28-41
12. Davis L (ed) (1991) Handbook of Genetic Algorithms. Van Nostrand Reinhold, NY
13. Eiben AE, Hinterding R, Michalewicz Z (1999) Parameter Control in Evolutionary Algorithms. IEEE Transactions on Evolutionary Computation 3(2): 124-141
14. Gruau F, Whitley D (1993) Adding Learning to the Cellular Development of Neural Networks: Evolution and the Baldwin Effect. Evolutionary Computation 1(3):213-233
15. Gorges-Schleuter M (1990) Genetic Algorithms and Population Structures - A Massively Parallel Algorithm. Dissertation, Dept. Comp. Science, University of Dortmund, Germany
16. Gorges-Schleuter M (1998) A Comparative Study of Global and Local Selection in Evolution Strategies. In: Eiben AE, Bäck T, Schoenauer M, Schwefel H-P (eds): Conf. Proc. PPSN V, LNCS 1498, Springer, Berlin, pp 367-377
17. Gorges-Schleuter M (1999) An Analysis of Local Selection in Evolution Strategies. In: Banzhaf W et al. (eds): Conf. Proc. GECCO 1999, Morgan Kaufmann, San Francisco, pp 847-854

18. Hart WE (1994) Adaptive Global Optimization with Local Search. Dissertation, University of California, USA
19. Hart WE, Krasnogor N, Smith JE (2004) Editorial introduction. Evolutionary Computation 12(3):v-vi (special issue on Memetic Algorithms)
20. Hart WE, Krasnogor N, Smith JE (Eds.) (2004) Recent Advances in Memetic Algorithms. Studies in Fuzziness and Soft Computing, vol. 166, Springer
21. Hansen, B (2006) Compilation of Results on the 2005 CEC Benchmark Function Set. CoLab, Institute of Computational Science, ETH Zurich, Tech. Rep. Or: http://www3.ntu.edu.sg/home/epnsugan/index_files/CEC-05/compareresults.pdf Accessed 18 August 2009
22. Jakob W (2002) HyGLEAM – An Approach to Generally Applicable Hybridization of Evolutionary Algorithms. In: Merelo JJ, et al. (eds.): Conf. Proc. PPSN VII, LNCS 2439, Springer, Berlin, pp 527–536
23. Jakob W, Blume C, Bretthauer G (2004) Towards a Generally Applicable Self-Adapting Hybridization of Evolutionary Algorithms. In: Conf. Proc. GECCO 2004, LNCS 3102, Springer, Berlin, pp 790-791 and LBP
24. Jakob W (2006) Towards an Adaptive Multimeme Algorithm for Parameter Optimisation Suiting the Engineers' Needs. In: Runarsson TP et al. (eds.): Conf. Proc. PPSN IX, LNCS 4193, Springer, Berlin, pp 132-141
25. Jakob W (2008) A Cost-Benefit-Based Adaptation Scheme for Multimeme Algorithms. In: Wyrzykowski et al. (eds): Conf. Proc. Parallel Processing and Applied Mathematics 2007, LNCS 4967, Springer, Berlin, pp 509-519
26. Jakob W, Quinte A, Stucky K-U, Süß W (2008) Fast Multi-objective Scheduling of Jobs to Constrained Resources Using a Hybrid Evolutionary Algorithm. In: Rudolph G (eds.): Conf. Proc. PPSN X, LNCS 5199, Springer, Berlin, pp 1031-1040
27. Krasnogor N, Smith JE (2001) Emergence of Profitable Search Strategies Based on a Simple Inheritance Algorithm. In: Conf. Proc. GECCO 2001, M. Kaufmann, San Francisco, pp 432-439
28. Krasnogor N, Blackburne BP, Burke EK, Hirst JD (2002) Multimeme Algorithms for Protein Structure Prediction. In: Conf. Proc. PPSN VII, LNCS 2439, Springer, Berlin, pp 769-778, citation: p.772
29. Krasnogor N (2002) Studies on the Theory and Design Space of Memetic Algorithms. Dissertation, University West of England, UK
30. Krasnogor N, Smith J (2005) A Tutorial for Competent Memetic Algorithms: Model, Taxonomy, and Design Issues. IEEE Transactions on Evolutionary Computation (9)5:474-488
31. Law NL, Szeto KY (2007) Adaptive genetic algorithm with mutation and crossover matrices. In: Veloso MM (ed) Conf. Proc. IJCAI-07, pp 2330-2333
32. Land MWS (1998) Evolutionary Algorithms with Local Search for Combinatorial Optimization. Dissertation, University of California, USA

33. Lozano M, Herrera F, Krasnogor N, Molina D (2004) Real-Coded Memetic Algorithms with Crossover Hill-Climbing. Evolutionary Computation 12(2):273-302
34. Meuth R, Lim MH, Ong YS, Wunsch II DC (2009) A proposition on memes and meta-memes in computing for high-order learning. J Memetic Comp. 1(2):85-100
35. Nelder JA, Mead RA (1965) Simplex Method for Function Minimization. The Computer Journal, 7, pp 308-313
36. Neri F, Tirronen V, Kärkkäinen T, Rossi T (2007) Fitness Diversity Based Adaptation in Multimeme Algorithms: A Comparative Study. In: Conf. Proc. IEEE Congr. On Evol. Comp. CEC 2007, pp 2374-2381
37. Neri F, Toivanen J, Cascella GL, Ong YS (2007) An Adaptive Multimeme Algorithm for Designing HIV Multidrug Therapies. IEEE/ACM Trans. Comput. Biol. Bioinformatics 4(2):264-278
38. Neri F, Tirronen V (2008) On memetic Differential Evolution frameworks: A study of advantages and limitations in hybridization. In: Conf. Proc. IEEE Congr. On Evol. Comp. CEC 2008, pp 2135-2142
39. Neri F, Tirronen V (2009) Scale factor local search in differential evolution. J Memetic Comp. 1(2):153-171
40. Nguyen QH, Ong YS, Krasnogor N (2007) A Study on Design Issues of Memetic Algorithm. In: Conf. Proc. IEEE Congr. on Evol. Comp. CEC 2007, pp 2390-2397
41. Noman N, Iba H (2008) Accelerating Differential Evolution Using an Adaptive Local Search. IEEE Trans. on Evol. Comput. 12(1):107-125
42. Ong YS, Keane AJ (2004) Meta-Lamarckian Learning in Memetic Algorithms. IEEE Trans. on Evol. Comput. 8(2):99-110
43. Ong YS, Lim MH, Zhu N, Wong KW (2006) Classification of Adaptive Memetic Algorithms: A Comparative Study. IEEE Trans. Systems, Man, and Cybernetics, Part B 36(1):141-152
44. Orvosh D, Davis L (1993) Shall We Repair? Genetic Algorithms, Combinatorial Optimization, and Feasibility Constraints. In: Forrest S (ed): Conf. Proc. Fifth ICGA, Morgan Kaufmann, San Mateo, CA, p 650
45. Posik P (2005) Real-parameter optimization using the mutation step co-evolution. In: Conf. Proc IEEE Congr. on Evol. Comp. CEC 2005, pp 872-879
46. Rosenbrock HH (1960) An Automatic Method for Finding the Greatest or Least Value of a Function. The Computer Journal, 3, pp 175-184
47. Schwefel H-P (1995) Evolution and Optimum Seeking. John Wiley & Sons, New York
48. Smith JE (2003) Co-evolving Memetic Algorithms: A learning approach to robust scalable optimisation. In: Conf. Proc. IEEE Congr. on Evol. Comp. CEC 2003, pp 498-505
49. Smith JE (2008) Self-Adaptation in Evolutionary Algorithms for Combinatorial Optimisation. In: Cotta C et al. (eds) Adaptive and Multilevel Metaheuristics, SCI 136, Springer, Berlin, pp 31-57
50. Sudholt D (2009) The impact of parameterization in memetic evolutionary algorithms. Theoretical Computer Science, DOI:10.1016/j.tcs.2009.03.003

51. Suganthan PN, Hansen N, Liang JJ, Deb K, Chen Y-P, Auger A, Tiwari S (2005) Problem Definition and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization. Technical Report, Nanyang Techn. University, Singapore and KanGAL Technical Report #2005005, IIT Kanpur, India. Available at http://www.ntu.edu.sg/home/epnsugan
52. Szeto KY, Zhang J (2005) Adaptive genetic algorithm and quasi-parallel genetic algorithm: Application to low-dimensional physics. In: Conf. proc. Large-Scale Scientific Computing (LSSC 2005), LNCS 3743, Springer, Berlin, pp 189-196
53. Takahara S, Kusumoto Y, Miyamoto S (2001) An adaptive meta-heuristic approach using partial optimization to non-convex polygons allocation problem. In: Conf. Proc. Fuzzy Systems 2001, IEEE, vol. 3, pp 1191-1194
54. Tirronen V, Neri F (2009) Differential Evolution with Fitness Diversity Self-adaptation. In: Chiong R (ed) Nature-inspired Algorithms for Optimisation, SCI 193, Springer, Berlin, pp199-234
55. Whitley D, Gordon V, Mathias K (1994) Lamarckian Evolution, The Baldwin Effect and Function Optimization. In: Davidor Y et al. (eds): Conf. Proc. PPSN III, LNCS 866, Springer, Berlin, pp 6-14
56. Yang J-M, Kao C-Y (2000) Integrating adaptive mutations and family competition into genetic algorithms as function optimizer. Soft Computing 4(2):89-102
57. Zitzler E, Teich J, Bhattacharyya SS (2000) Optimizing the Efficiency of Parameterized Local Search within Global Search: A Preliminary Study. In: Conf. Proc IEEE Congr. on Evol. Comp. CEC 2000, pp 365-372