Adaptive Cellular Memetic Algorithms

Nguyen Quang Huy nguy0046@ntu.edu.sg
School of Computer Engineering, Nanyang Technological University, 639798, Singapore

Ong Yew Soon asysong@ntu.edu.sg
School of Computer Engineering, Nanyang Technological University, 639798, Singapore

Lim Meng Hiot emhlim@ntu.edu.sg
School of Electrical and Electronic Engineering, Nanyang Technological University, 639798, Singapore

Natalio Krasnogor Natalio.Krasnogor@nottingham.ac.uk
School of Computer Science, Jubilee Campus, University of Nottingham, Nottingham, NG8 1BB, United Kingdom

Abstract
A Cellular Genetic Algorithm (CGA) is a decentralized form of GA where individuals in a population are usually arranged in a 2-D grid and interactions among individuals are restricted only within a set neighborhood. In this paper, we extend the notion of cellularity to Memetic Algorithms (MA), a configuration termed Cellular Memetic Algorithm (CMA). In addition, we propose adaptive mechanisms that tailor the amount of exploration versus exploitation of local solutions carried out by the CMA. We systematically benchmark this adaptive mechanism and provide evidence that the resulting adaptive CMA outperforms other methods both in the quality of solutions obtained and the number of function evaluations for a range of continuous optimization problems.

Keywords
Memetic Algorithm, Parallel Genetic Algorithm, Cellular Genetic Algorithm, Cellular Memetic Algorithm

1 Introduction

Evolutionary algorithm (EA) serves as appealing optimisation strategy for solving non-linear programming problems characterized with non-convex, disjoint and noisy solution spaces. Unlike conventional numerical optimization methods, evolutionary algorithms such as Genetic algorithm (GA), produce new design points that do not use information about the local slope of the objective function and are thus not prone to stalling at local optima. GA was inspired by Darwinian's survival of the fittest strategy with sexual reproduction, and Mendel's theory of genetics as the basis of biological inheritance. GA maintains a population of solutions; making use of competitive selection, recombination and mutation operators to generate new solutions which are biased towards better regions of the search space. Their popularity also lies in the ease of implementation and their ability to converge close to the global optimum. Although relatively easy to implement, a naive GA will usually exhibit one (if not both) of the following problems: (a) very slow convergence to local optima; (b) rapid diversity loss. Both of these problems often limits the practicality of GAs on many complex real world problems where computational time is a crucial consideration. Hence, it is now well established that "canonical" GAs are not suited for fine-tuning in complex search spaces, and that hybridization with other local search techniques can greatly improve the efficiency of search (Ong and Keane, 2004; Krasnogor and Smith, 2005).

The Memetic Algorithm (MA) is population-based meta-heuristic search method inspired by both Darwinian principles of evolution by natural selection and Dawkins' notion of a meme defined as a unit of cultural evolution that is capable of local refinements. That is, a memetic model for optimisation exhibits the plasticity of individuals that a strictly genetic model fails to capture. Recent studies on MAs have revealed their successes on a wide variety of optimization problems (Burke et al., 2001; Ishibuchi et al., 2003; Hart et al., 2004b; Lim and Xu, 2005; Tang et al., 2007; Hasan et al., 2009). Theoretical and empirical analyses (Houck et al., 1997; Digalakis and Margaritis, 2000; Ong and Keane, 2004; Krasnogor and Smith, 2005, 2008; Yu et al., 2009) have shown that MAs not only converge to high quality solutions, but also search more efficiently than their conventional counterparts. In a more diverse context, MAs are also commonly known as hybrid EAs, Baldwinian EAs, Lamarkian EAs, cultural algorithms and genetic local search. For an overview on the recent advances in the field, the reader is referred to two special issues on MA (Hart et al., 2004a; Ong et al., 2007, 2008).

An evolutionary algorithm (EA) without any structure is usually referred to as a panmictic EA and the large majority of Memetic Algorithm have been implemented in this fashion. On the other hand, a popular way of structuring the population is by recurring to a lattice-like topology in which individual interact only with their nearest neighbors. A cellular GA (CGA) is a prime exemplar of this technique (Martin et al., 1997; Pettey, 1997). Structured GAs, not only favor a parallel implementation due to the decentralized management of the populations, but also they are generally known to provide better sampling of the search space and therefore perform well in many cases (Bradwell et al., 1999; Alba and Troya, 2002; Tang et al., 2006). Earlier works (Muhlenbein et al., 1991; Baluja, 1993) have demonstrated the efficacy of CGA for complex optimization tasks. The small overlap of neighborhoods and spatial structure of CGA induces a slow diffusion of information throughout the whole population. This has the advantage of preventing, to a large extend, premature convergence, i.e. problem (b) mentioned above. The drawback of CGAs, however, is that significantly greater effort is often required before converging to near global optimum. In this respect, CGA has great potential for performance enhancement by equipping it with better exploitation capabilities. Thus the synergy between CGA and local search procedures was investigated by Folino et al. in (Folino et al., 1998). By combining CGA with a random walk local search, the authors achieved better convergence rate on the satisfiability problems. Later, Alba et. al. (Alba et al., 2005) formalized this class of algorithms under a framework named cellular memetic algorithm (CMA). The work of Alba, (Alba et al., 2005), among many others such as Hart (1994); Goldberg and Voessner (1999); Krasnogor and Smith (2000, 2005); Ong et al. (2006); Krasnogor and Smith (2008), have highlighted some important design issues that must be taken into consideration when designing hybrid evolutionary-local search algorithms, including CMA.

In this paper, the focus is on the incorporation of suitable local improvement mechanisms for enhancing the evolutionary search efficiency of CGA. In particular, we begin with a study on the canonical Cellular MA or CMA in short. To facilitate a healthy balance in exploration (global search) and exploitation (local improvement) under a fixed computational budget, we proposed an adaptive Cellular MA. In contrast to the canonical MAs and CGAs, the Adaptive Cellular MA (ACMA) contains autonomic mechanisms for controlling the local search frequency, i.e., how often should local learning be applied at each stage of the algorithm, and defining which individuals in the population will undergo local improvements. In ACMA, adaptation is carried out based on the distribution or diversity of a population. In particular, we use fitness values to estimate the level of similarity between individuals and as a basis for applying local search on different groups of individuals selectively, thus offering ease of implementation. This work thus generalises and extends (Krasnogor and Smith, 2000) in its use of fitness diversity as a control "knob" for tuning the exploration/exploitation balance in CMA. Empirical studies on both CMA and ACMA are conducted using a series of commonly used benchmark test functions. Results obtained show that ACMA converges to competitive solutions at significantly lower computational cost compared to the canonical CMA and MA. Furthermore, ACMA is shown to display better robustness in terms of solution quality on the problems considered.

This paper is organized in the following manner. Section II presents a brief overview of the Cellular GA. The canonical Cellular MA and the proposed Adaptive CMA are presented in section III. Section IV summarizes our empirical studies on continuous parametric benchmark functions and provides a comprehensive quantitative and statistical comparison of CGA, multistart local search, CMA and ACMA. The search performances of the various algorithms in terms of solution quality, computational effort, and solution precision are also reported in the section followed by analysis of the results obtained. Finally, Section V concludes the paper.

2 Cellular Genetic Algorithm

In the Cellular GA, a lattice structure is used to represent the neighborhood relation between individuals in the population. Individuals can only exchange information, for example through crossover or other cooperative strategy, with other individuals constrained within a small and localised region of this lattice structure (see Figure 1). These partially overlapped small neighborhoods, which provide structure to CGA population, help to promote the preservation of diversity (Spiessens and Manderick, 1991) and hence facilitate a more robust exploration of the search space. Exploitation, on the other hand, takes place within each neighborhood through the usual genetic operators. In this paper we we focus on the two-dimensional grid, one of the most commonly used CGA's topologies. As illustrated in Figure 1, a local neighborhood is made up of five individuals, i.e., the individual considered at grid position (x, y), together with its immediate neighbors in north, east, west, and south orientations. This lattice is commonly known as the Von-Neumann neighborhood.

The pseudo-code of the CGA is outlined in Algorithm 1. Here, we consider the general optimization problem of finding the *global minimum* $\hat{\mathbf{x}}$ such that $\hat{\mathbf{x}} = \underset{x}{\mathrm{argmin}} f(\mathbf{x})$, where $\mathbf{x}$ is the set of design variables. In the CGA, each individual undergoes the standard genetic operators including crossover and mutation. In contrast to the canonical GA -as mentioned above- an individual can only mate with partner(s) in the Von-Neumann neighborhood during reproduction. Subsequently, the fitness value of the offspring is computed. The offspring then replaces the original individual at its grid location if it results in a more favorable fitness value.

As the bias towards better solutions is handled locally, the resulting lower selection pressure of the CGA makes it converge at a slower rate than a panmictic GA. For exam-

Algorithm 1 Cellular Genetic Algorithm

```
1: procedure CellularGA
2:     $pop$ = Create-Grid(($WIDTH * HEIGHT$))
3:     for $x = 1$ to $WIDTH$ do
4:         for $y = 1$ to $HEIGHT$ do
5:             Initialize $pop(x, y)$
6:             $pop(x, y)$.fitness = Evaluate($pop(x, y)$)
7:         end for
8:     end for
9:     while (termination condition is not satisfied) do
10:        $oldpop = pop$
11:        for $x = 1$ to $WIDTH$ do
12:            for $y = 1$ to $HEIGHT$ do
13:                $parent1$ = $oldpop(x, y)$
14:                $parent2$ = Select(Neighbors($x, y$))
15:                $child$ = Crossover($parent1, parent2$)
16:                Mutate($child$)
17:                $child$.fitness = Evaluate($child$)
18:                if ($child$.fitness $<$ $oldpop(x, y)$.fitness) then
19:                    $pop(x, y) = child$
20:                end if
21:            end for
22:        end for
23:    end while
24: end procedure
```

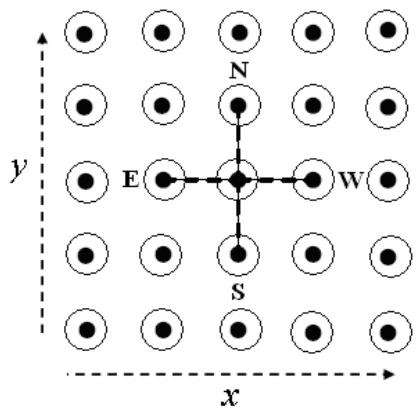

Figure 1: Cellular GA neighborhood structure

ple, a parallel CGAs (fine-grained) for flow shop scheduling problems has been shown to converge slower than a parallel island model GA and to generate better solution quality (Kohlmorgen et al., 1999). Alba et al. (2002) investigated various population update policies for CGA and the effect of such policies on the tradeoff between exploration and exploitation during the search process. The effect of grid structure on the performance of CGA was also studied in (Alba and Dorronsoro, 2005). Across the large majority of research results on the use of CGA, the ability to reduce the effects of premature convergence are counterweighted by the fact that this usually leads to a slow convergence rate for the underlying search process. In this paper we are tackling precisely this challenge, namely, to demonstrate a mechanism by which both diversity and fast convergence can be harmonised en route to solving optimization problems that are computationally demanding.

3 Cellular Memetic Algorithm

In this section, we begin with the description of a canonical CMA. An efficient adaptive CMA is then proposed for effective selection of suitable candidates in the CMA population to undergo local search improvement in each search generation. By coupling local search with the usual genetic operators present in a CGA we aim at improving the convergence speed of the method.

3.1 Canonical Cellular Memetic Algorithm

The pseudo code for a canonical CMA is outlined in Algorithm 2. For the sake of brevity, we present the canonical CMA as two separate phases. Phase I involves the CGA and is hence similar to that of Algorithm 1. After phase I, the offspring undergoes local search. To avoid confusion, it is worth noting that two concepts of locality (neighborhood) exist in the CMA. Neighborhoods can be defined in the context of population grid structures or that of the solution space. On the one hand, the local improvement procedures operate on the search space itself, i.e., around the vicinity of the incumbent solution. On the other hand, the selection of neighboring individuals takes place through reference to lattice structure upon which the population is mapped, in this case, the Von-Neumann neighborhood. Since they refer to a totally different neighborhood conceptions, individuals which are neighbors in the lattice (i.e. the population structure) do not necessarily represent neighboring search space points, i.e., they may very well be far in terms of genotypic, phenotypic and/or fitness distance.

Algorithm 2 Cellular Memetic Algorithm

```
procedure CELLULARMA
    Initialize-Meme-Pool;
    $pop$ = Create-Grid(($WIDTH * HEIGHT$))
    for $x = 1$ to $WIDTH$ do
        for $y = 1$ to $HEIGHT$ do
            Initialize $pop(x, y)$
            $pop(x, y)$.fitness = Evaluate($pop(x, y)$)
        end for
    end for
    while (termination condition is not satisfied) do
        $generation = generation + 1$
        $oldpop = pop$
        for $x = 1$ to $WIDTH$ do
            for $y = 1$ to $HEIGHT$ do
                /* Phase I: CGA */
                $parent1$ = $oldpop(x, y)$
                $parent2$ = Select(Neighbors($x, y$))
                $child$ = Crossover($parent1, parent2$)
                $child$ = Mutate($child$)
                /* Phase II: Individual Learning */
                if ($generation \mod \theta = 1$) then
                    $child$ = Local-Improvement($child$)
                end if
                $child$.fitness = Evaluate($child$)
                if ($child$.fitness $<$ $oldpop(x, y)$.fitness) then
                    $pop(x, y) = child$
                end if
            end for
        end for
    end while
end procedure
```

3.2 Adaptive Cellular Memetic Algorithm

The canonical CMA employs local search on all individuals of the population. While the canonical model fully exploits all individuals in the population, this is a very computationally intensive and inefficient search process. At the same time, exhaustive local search may lead to ineffective search due to premature drop in diversity during the MA search. Recent study on the effect of local search frequency or probability of local search, i.e., how many chromosomes to undergo local search in an EA generation, highlights the need for adaptation of this parameter in order to attain good MA search performance (Hart, 1994; Deb and Beyer, 2001). In the following subsections, we study two schemes for selecting the individuals undergoing local search. Based on the taxonomy in (Ong et al., 2006), both algorithms represent forms of static adaptation approach.

To help categorize the new algorithms we propose in relation to other Memetic Algorithms, please note that accordingly to the taxonomy developed in (Ong et al., 2006), both our methods represent a "static adaptation approach". In the complementary taxonomy described in Krasnogor and Smith (2005), these methods have an index $D = 4$ as they employ a "coarse grain scheduler" that, using both global and local population information, allocates local search trials. For more details the reader is referred to the above cited references.

- Cellular Memetic Algorithm with random selection scheme (CMAR)

  The most basic scheme for selecting the subset of chromosomes that will undergo the improvement procedure is random selection. In the local search phase, chromosomes are randomly selected to undergo local search - this does not adapt but has the advantage of giving all the chromosomes in the CMA population an equal chance to undergo improvement.

- Stratified-Adaptive Cellular Memetic Algorithm (ACMA)

  An alternative scheme for selecting suitable individuals to undergo local search during the CMA run is based on the online population statistics. In particular, adaptation is carried out based on the distribution or diversity measure of the population. The diversity of a GA population may be measured by various means that can be applied at the genotype, phenotype or fitness level. The reader is referred to Burke et al. (2002, 2003) for a detailed discussion of these issues with an emphasis on Genetic Programming but with general application to other EAs.

Our approach maintains appropriate degree of local search and solution diversity in the evolutionary search based on a fitness uniform selection scheme. The adaptive local search scheme is outlined in Algorithm 3. In ACMA, we use fitness values to estimate the level of similarity between individuals and as a basis for applying local search on different groups of individuals selectively. At each generation, using the fitness range in the population, i.e., the maximum and minimum fitness values of the current CGA population, the fitness space is divided into $\alpha * n$ number of uniform intervals, where α is a user-predefined percentage value of population undergoing local refinement while n is the population size, i.e., $WIDTH \times HEIGHT$. This is referred to as $fitnessrange$ in Algorithm 3.

Subsequently, each individual in the population is then assigned to one of the $\alpha * n$ groups based on its fitness value. Henceforth, only one individual from each group undergoes local search. This individual may be selected either randomly or through some other selection schemes. In this manner, the likelihood of applying local search on similar (fitness-wise) individuals is reduced. Here we also consider performing local search on the elite individual, since it is most likely to be near the best optimum solution. To avoid redundant local searches, a black-list of past individuals that did not benefit from local improvement procedure is maintained during the search.

In summary, the number of local searches made in each search generation is constrained to a maximum of $\alpha * n + 1$, i.e., the number of fitness groups plus the elite individual. Note that the actual number of local searches may differ between two generations, as it is possible that none of the individuals has a fitness value that falls under a particular fitness bin. That is, fitness groups may be empty hence no sample from within that group can be used to perform local search . In this way, ACMA

Algorithm 3 Adaptive Local Search

```
if $generation \mod \theta = 1$ then
    /* Initialization will create $\alpha * n$ empty groups */
    $fitnessrange = (maxfitness - minfitness)/(\alpha * n)$
    if ($fitnessrange > \varepsilon$) then
        for each individual $indv$ in the current population $pop$ do
            $groupindex = (indv.\text{fitness} - minfitness)/fitnessrange$
            assign $indv$ to a group based on the $groupindex$
        end for
        for each non-empty groups $g$ do
            $i$ = Random(1, size of $g$)
            if (g[i] is not in black-list) then
                $childls$ = Local-Improvement ($g[i]$)
                if ($childls$.fitness $<$ $g[i]$.fitness) then
                    $g[i] = childls$
                else
                    Add $g[i]$ to black-list
                end if
            end if
        end for
    else/* $fitnessrange < \varepsilon$ */
        re-initialize the entire population except the elite individuals
    end if
end if
```

performs exploration through its neighborhood structure while at the same time exploitation is carried out through local search which, in turn, adapts as a function of the grouping in fitness space. In the design of memetic algorithms, it is common knowledge that a good balance between exploitation and exploration is the key to achieving good search performance. Hence, to maintain good diversity and avoid premature convergence, random individuals are introduced into the population whenever the search exhibits signs of premature convergence. Here, we favor greater exploration by introducing new members into the population when $fitnessrange < \varepsilon$, see $fitnessrange = (maxfitness - minfitness)/(\alpha * n)$ in Algorithm 3, since premature convergence suggests that excessive exploitation might have occurred during the search process. ε is a user-specific parameter, which denotes some small value. This means that when the fitness range of values between successive groups is small enough, individuals have become very similar in the fitness or solution space. Therefore, new search points are introduced into the next generation to improve the diversity while the best individual is retained.

Further, we illustrate the selective local search mechanism in the proposed ACMA using the Rastrigin benchmark test function. The results compiled are shown in Figs 2-4. In particular, Figure 2 shows a plot of the fitness distribution at 60 generations when using the proposed ACMA to optimize the 10-D Rastrigin function. With α configured as 5% and a population size n of 100, the number of fitness groups is thus $\alpha * n = 5$. The $fitnessrange$ of each fitness group is then approximately 2.2995. From each group, one individual is picked randomly to undergo local search. This results in a total of six local searches made at generation 60, i.e., one local search for each fitness group and

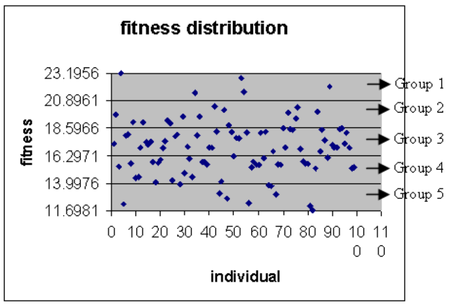

Figure 2: Fitness distribution at generation 60 (minimizing 10D Rastrigin)

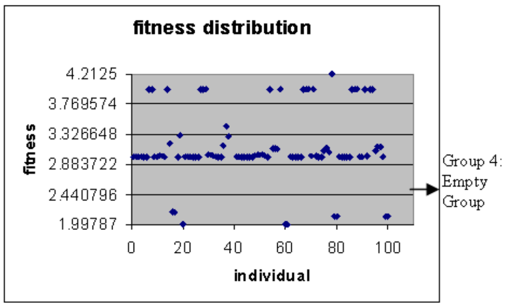

Figure 3: Fitness distribution at generation 122 (minimizing 10D Rastrigin)

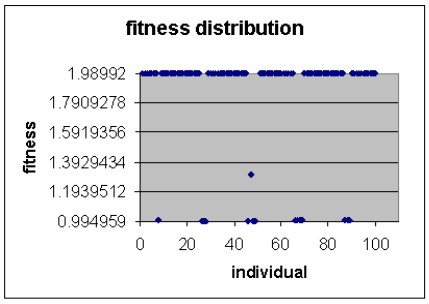

Figure 4: Fitness distribution at generation 240 (minimizing 10D Rastrigin)

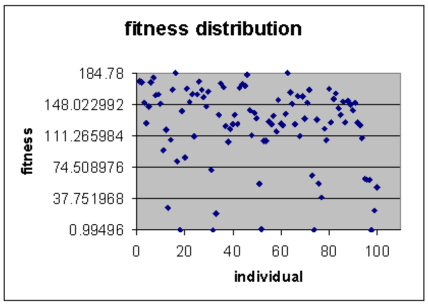

Figure 5: Fitness distribution at generation 266 (minimizing 10D Rastrigin)

another on the elite individual.

At generation 122, it is possible to observe in Figure 3 that some form of convergence is emerging, a sign of possible over-exploitation. The population of individuals at generation 122 are similar in solution or fitness space, particularly there are many individuals having similar fitness values in groups 1 and 3. The $fitnessrange$ is now thus smaller than before, approximately 0.442926. Note that no individuals are assigned to the group in $fitnessrange$ of $[2.440796, 2.883722]$ or group 4, since none of the individuals in the population has a fitness value that falls within this range. The total number of local searches performed is thus at most five (one on each non-empty fitness grouping and another on the elite individual). At generation 240, the population diversity can be observed to have dropped significantly in Figure 4 with most individuals converging at local optima with fitness levels 0.994959 and 1.98992. There are now two empty groups and the number of local searches has decreased to at most four. To summarize, Figures 2 - 5 illustrate how ACMA algorithm adapts the amount of exploitation by adapting the number of local searches throughout its search process. As the search continues, the $fitnessrange$ shrinks and individuals in the population tend to be very similar to each other in the fitness space. Here, we configure ε to a very small value of 10^{-8} in our illustration. When the condition for $fitnessrange < \varepsilon$ is satisfied, new random individuals are introduced into the population to replace all except the elite individual in an effort to maintain good diversity and solution quality in the search.

Figure 5 shows the fitness distribution when new random search points are introduced at generation 266, but with the elite individual of fitness 0.99496 preserved in the population. Note that the number of local searches has been restored back to at most six and the process repeats until the termination condition is satisfied.

4 Empirical study

In this section, we present a numerical study on CGA, canonical CMA and adaptive CMAs in optimizing a series of commonly used benchmark functions used in the literature. The search performance results of a stochastic multi-start local search (label here as MS-DSCG in short) is also reported as the baseline for which other stochastic algorithms may be compared. Here, the strategy of Davies, Swann, and Campey with Gram-Schmidt orthogonalization (DSCG) is used as the local search procedure as it has been shown, e.g. (Ong and Keane, 2004), that it generates good solution for continuous domains in which MAs are used. In particular all the metaheuristics implemented here, namely, the multi-start local search, canonical CMA, CMAR and the ACMA will have DSCG as their core local searcher.

In the DSCG procedure, the search begins with starting point $\mathbf{x}^{(k,i)} = \mathbf{x}^{(1,0)}$, where i and k are the direction and iteration counters, respectively. A line search is conducted on each independent n dimension along direction set $\{\mathbf{v}^{(k)}\}$. At the end of each iteration k, a new set of orthogonal search directions, $\{\mathbf{v}^{(k+1)}\}$, is generated. This process is repeated until convergence to a local optimum or the allowable computational budget has elapsed. For the details, the readers are referred to Figure 6 which outlines the main steps of the DSCG procedure.

In the next subsection, we study the utility of the adaptive local search mechanism introduced in the CGA and analyze the search performance results obtained by the algorithms on the benchmark problems.

4.1 Experimental Study

We performed twenty five independent runs of each of Multi-start local search (MS-DSCG), Cellular GA (CGA), canonical Cellular MA (CMA), Cellular MA with random selective local search $\alpha = 5\%$ (CMAR5), Adaptive Cellular MA with $\alpha = 5\%$ (ACMA5), and Adaptive Cellular MA with $\alpha = 10\%$ (ACMA10).

The algorithms are tested on the thirty-dimensional parametric optimization problems already extensively discussed in the literature (Digalakis and Margaritis, 2001) and a real-world problem, the Frequency Modulation Sound (FMS) function (Tsutsui and Fujimoto, 1993). Table 1 tabulates these benchmark test functions with their notable characteristics. Our main purpose is to compare the performance of ACMA with CGA and CMA while the MS-DSCG is considered as the base-line. We present results for ACMA with $\alpha = 5\%$ and 10% to observe the effect of different exploitation level on search performance.

In the present study, the evolutionary parameters for all the algorithms are configured consistently with population size of 100, bit-flip mutation rate of 0.01, two-point crossover rate of 0.9 and 32-bit binary encoding for each search variable. In all the MA variants considered, the local search frequency, labeled as θ in Algorithm 2, is set to 10 generations, i.e., local search phase is applied for every 10 generations. In each local search phase, the initial step size, $s^{(0)}$, is configured to 1, while a maximum computational budget of 300 evaluations for a single individual refinement process is considered.

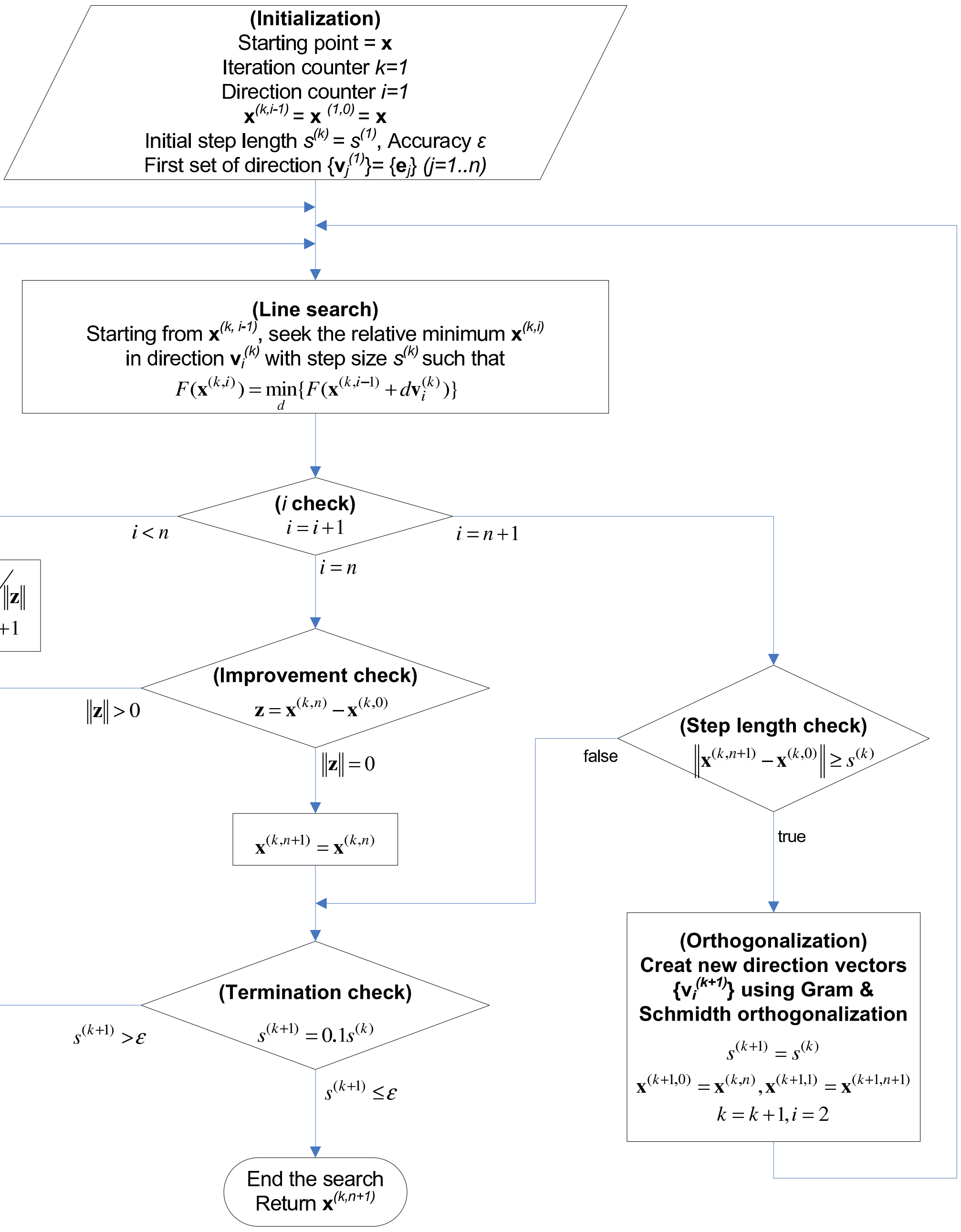
(Initialization)
Starting point = $\mathbf{x}$
Iteration counter $k=1$
Direction counter $i=1$
$\mathbf{x}^{(k,i-1)} = \mathbf{x}^{(1,0)} = \mathbf{x}$
Initial step length $s^{(k)} = s^{(1)}$, Accuracy ε
First set of direction $\{\mathbf{v}_j^{(1)}\} = \{\mathbf{e}_j\}$ $(j=1..n)$
$\mathbf{x}^{(k+1,0)} = \mathbf{x}^{(k,n+1)}$ $k=k+1$ $i=1$
(Line search)
Starting from $\mathbf{x}^{(k,i-1)}$, seek the relative minimum $\mathbf{x}^{(k,i)}$ in direction $\mathbf{v}_i^{(k)}$ with step size $s^{(k)}$ such that
$F(\mathbf{x}^{(k,i)}) = \min_d\{F(\mathbf{x}^{(k,i-1)} + d\mathbf{v}_i^{(k)})\}$
(i check)
$i=i+1$
$i<n$
$i=n+1$
$i=n$
$\mathbf{v}_{n+1}^{(k)} = \mathbf{z}/\|\mathbf{z}\|$ $i=n+1$
(Improvement check)
$\mathbf{z} = \mathbf{x}^{(k,n)} - \mathbf{x}^{(k,0)}$
$\|\mathbf{z}\|>0$
$\|\mathbf{z}\|=0$
$\mathbf{x}^{(k,n+1)} = \mathbf{x}^{(k,n)}$
(Step length check)
$\|\mathbf{x}^{(k,n+1)} - \mathbf{x}^{(k,0)}\| \geq s^{(k)}$
false
true
(Orthogonalization)
Creat new direction vectors $\{\mathbf{v}_i^{(k+1)}\}$ using Gram & Schmidth orthogonalization
$s^{(k+1)} = s^{(k)}$
$\mathbf{x}^{(k+1,0)} = \mathbf{x}^{(k,n)}, \mathbf{x}^{(k+1,1)} = \mathbf{x}^{(k+1,n+1)}$
$k=k+1, i=2$
(Termination check)
$s^{(k+1)} = 0.1s^{(k)}$
$s^{(k+1)} > \varepsilon$
$s^{(k+1)} \leq \varepsilon$
End the search
Return $\mathbf{x}^{(k,n+1)}$

Figure 6: The strategy of Davies, Swann, and Campey with Gram-Schmidt orthogonalization (DSCG).

Table 1: Benchmark functions used in the study (epi*: epistasis, mul*: multimodality)

Function	Range	Characteristics	
		Epi*	Mul*
$F_{Sphere} = \sum_{i=1}^{n} (z_i^2)$	$[-100, 100]^{30}$	none	none
$F_{Elliptic} = \sum_{i=1}^{n} (10^6)^{\frac{i-1}{n-1}}$	$[-100, 100]^{30}$	none	none
$F_{Schwefel1.2} = \sum_{i=1}^{n} (\sum_{j=1}^{i} z_j)^2$	$[-100, 100]^{30}$	none	none
$F_{Ackley}(\mathbf{x}) = 20 + e - 20e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)}$	$[-32, 32]^{30}$	weak	normal
$F_{Rastrigin}(\mathbf{x}) = 10n + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i))$	$[-5.12, 5.12]^{30}$	none	high
$F_{Griewank}(\mathbf{x}) = 1 + \sum_{i=1}^{n} x_i^2/4000 - \prod_{i=1}^{n} \cos(x_i/\sqrt{i})$	$[-600, 600]^{30}$	weak	high
$F_{Rosenbrock}(\mathbf{x}) = \sum_{i=1}^{n-1} (100 \times (x_{i+1} - x_i^2)^2 + (1 - x_i)^2)$	$[-100, 100]^{30}$	high	weak
$F_{Weierstrass} = \sum_{i=1}^{n} (\sum_{k=0}^{k_{max}} (a^k \cos(2\pi b^k (z_i + 0.5)))) - n \sum_{k=0}^{k_{max}} (a^k \cos(\pi b^k))$	$[-0.5, 0.5]^{30}$	weak	high
$F_{FMS}(a_1, w_1, a_2, w_2, a_3, w_3) = \sum_{t=0}^{100} (y(t) - y_0(t))^2$ $y(t) = a_1 \sin(w_1 \cdot t \cdot \theta + a_2 \sin(w_2 \cdot t \cdot \theta + a_3 \sin(w_3 \cdot t \cdot \theta)))$ $y_0(t) = 1.0 \sin(5.0 \cdot t \cdot \theta + 1.5 \sin(4.8 \cdot t \cdot \theta + 2.0 \sin(4.9 \cdot t \cdot \theta)))$ $\theta = 2\pi/100, \quad a_i, w_i \in [-6.4, 6.35], (i = 1, 2, 3).$	$[-6.4, 6.35]^{6}$	high	high
$F_{Scaffer} = \sum_{i=1}^{n} F(z_i, z_{i+1}), \quad z_{n+1} = z_1$ $F(x, y) = 0.5 + \frac{\sin^2(\sqrt{x^2 + y^2}) - 0.5}{(1 + 0.001(x^2 + y^2))^2}$	$[-100, 100]^{30}$	none	none

4.2 Results and Analysis

Figures 7-16 present the average convergence search trends of CGA, MS-DSCG, CMA, CMAR5, ACMA5, and ACMA10 across 50 independent simulation runs on each of the problems considered, respectively. Each simulation run continued until the global optimum is found or a maximum of $10000 * D$ (300,000 function evaluations in the present study) is reached, where is D is the problem dimensionality. The statistics of the search runs, i.e., the final errors with respect to the global optimum upon search termination, the number of evaluations, etc., are presented in Tables 2-11 to allow more detailed comparison on the solution quality and efficiency of the algorithms. In particular, if the algorithm reaches the global optimum (fitness error $< 10^{-8}$) before the maximum computational budget of 300,000 evaluations elapsed, the number of evaluations incurred in the search process will be reported. For instance, the best result displayed by CMA in Table 2 is reported as 0.0 (451). This implies the best run of CMA (across the 50 runs) is one that successfully converged to a fitness error of $< 10^{-8}$ from the global optimum at 451 evaluations.

For the sake of brevity, we present the discussion and analysis of our results divided into two main categories. The first category covers the unimodal problems, namely, Sphere, Elliptic and Schwefel 1.02 functions. The second category discussed the results obtained for the multimodal problem sets, in this case, Ackley, Rastrigin, Griewank, Rosenbrock, Weierstrass, Scaffer and FMS problems.

4.2.1 Unimodal functions (Sphere, Elliptic and Schwefel 1.02)

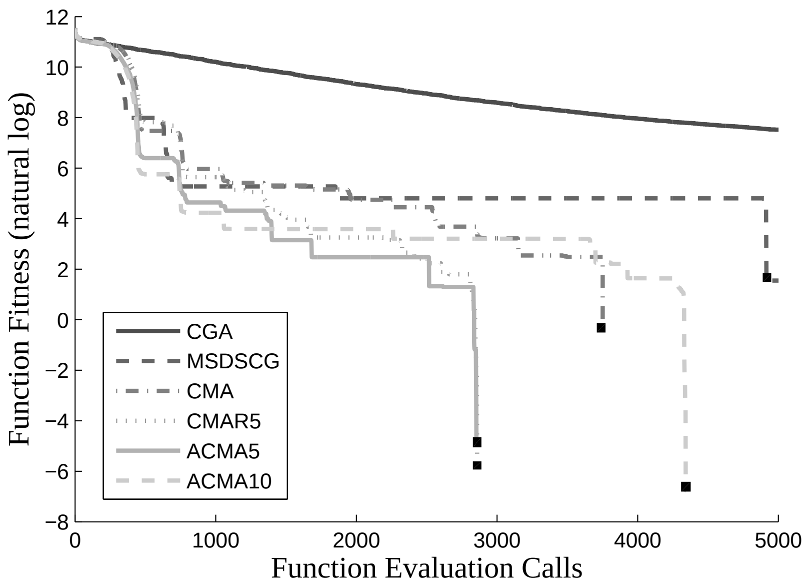

Figure 7: Search traces (average of 50 runs) for minimizing 10D Sphere function. (Square box indicates that the Global Optimum is found)

We begin first with a discussion on the results obtained for the unimodal problem set. The search traces of the different algorithms on Sphere and Elliptic functions are depicted in Figures 7 and 8. While CGA faces slow convergence on high dimen-

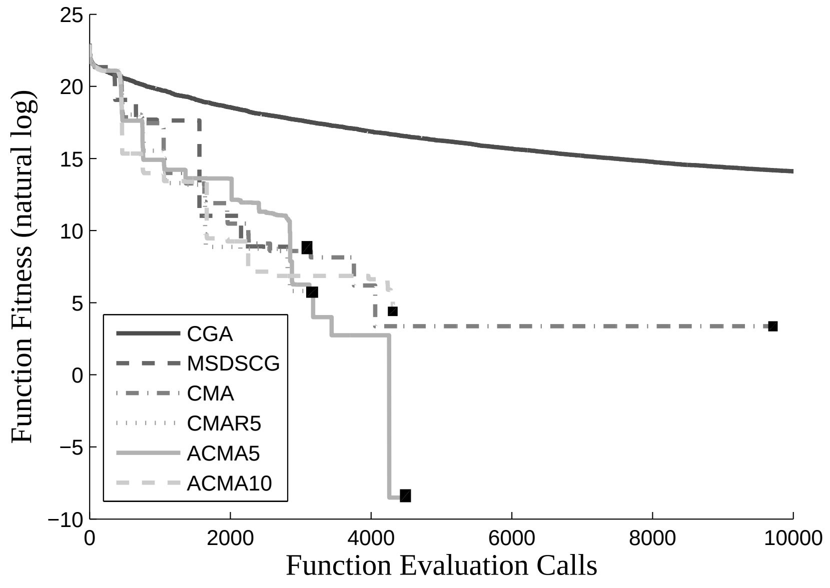

Figure 8: Search traces (average of 50 runs) for minimizing 30D Elliptic function. (Square box indicates that the Global Optimum is found)

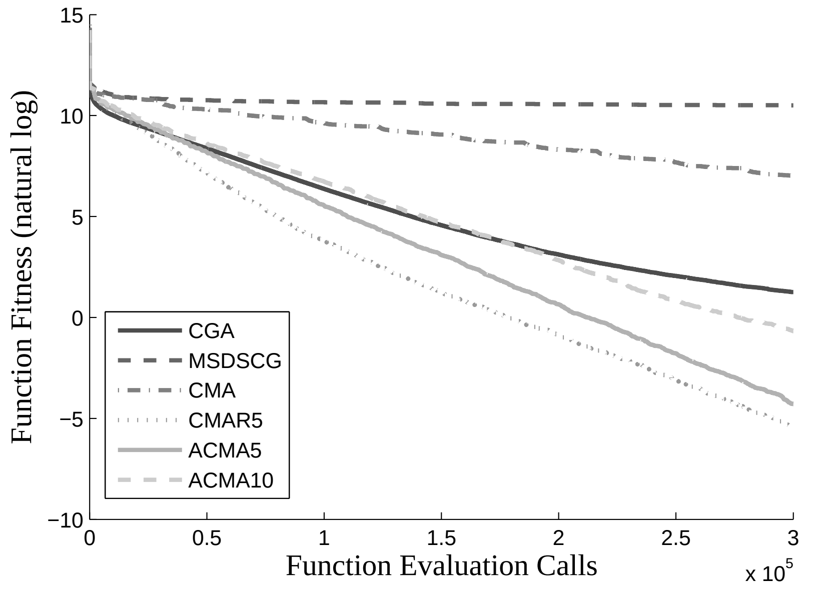

Figure 9: Search traces (average of 50 runs) for minimizing 30D Schwefel function.

Table 2: Fitness error values and Success Rate (30 Dimensional Sphere)(note that the best performing algorithm is highlighted in bold typeface)

Algorithm	Fitness error values and success rate				
	Best	Worst	Mean	Std	Success
CGA	0.000013	0.000081	0.000037	0.000014	0%
MS-DSCG	0.0 (344)	0.0 (6070)	0.0 (1365)	0.0 (1111)	100%
CMA	0.0 (451)	0.0 (3755)	0.0 (1227)	0.0 (852)	100%
CMAR5	0.0 (443)	0.0 (2862)	0.0 (910)	0.0 (552)	100%
ACMA5	0.0 (435)	0.0 (2857)	0.0 (742)	0.0 (603)	100%
ACMA10	**0.0 (429)**	**0.0 (4345)**	**0.0 (700)**	**0.0 (805)**	**100%**

Table 3: Fitness error values and success rate (30 Dimensional Elliptic)(note that the best performing algorithm is highlighted in bold typeface)

Algorithm	Fitness error values and success rate				
	Best	Worst	Mean	Std	Success
CGA	0.296993	11.802538	2.094388	2.013883	0%
MS-DSCG	0.0 (344)	0.0 (8191)	0.0 (1688)	0.0 (1686)	100%
CMA	0.0 (442)	0.0 (9753)	0.0 (1645)	0.0 (1494)	100%
CMAR5	0.0 (444)	0.0 (3135)	0.0 (1250)	0.0 (803)	100%
ACMA5	0.0 (448)	0.0 (4454)	0.0 (1675)	0.0 (1139)	100%
ACMA10	**0.0 (442)**	**0.0 (4312)**	**0.0 (1124)**	**0.0 (964)**	**100%**

Table 4: Fitness error values and success rate (30 Dimensional Schwefel 1.02)(note that the best performing algorithm is highlighted in bold typeface)

Algorithm	Fitness error values and success rate				
	Best	Worst	Mean	Std	Success
CGA	1.083250	6.511442	3.507798	1.379640	0%
MS-DSCG	25017.911043	46222.875190	36613.290116	4981.768773	0%
CMA	338.664096	1820.302667	1120.620191	332.701733	0%
CMAR5	**0.000302**	**0.015256**	**0.004470**	**0.003089**	**0%**
ACMA5	0.000694	0.084662	0.013748	0.016618	0%
ACMA10	0.017464	3.542390	0.509196	0.548611	0%

sional problems as expected, the remaining algorithms managed to locate the global optimum solution, thanks to inclusion of individual learning. The average number of evaluations reported in Table 2 and 3 highlights that the computational costs incurred by the CMA variants in converging to the global optimum accurately on Sphere and Elliptic functions are relatively close. Note that in all the CMA variants considered, the individual learning phase is conducted immediately after the first search generation. On the other hand, Sphere and Elliptic problems are unimodal and convex, thus DSCG is able to converge to the global optimum regardless of the starting point used. This also explains why the results obtained by a stochastic multistart individual learning is competitive to those obtained by MA.

In contrast, since the Schwefel function poses a challenge to the DSCG local search, MS-DSCG which is generates random starting points fails to search on the function well as shown in Figure 9. It is also observed that CMA does not fare as well as CGA, suggesting that performing local search on the entire population is inefficient. That also explains why CMAR5 and ACMA5 fare better than ACMA10.

4.2.2 Multimodal functions

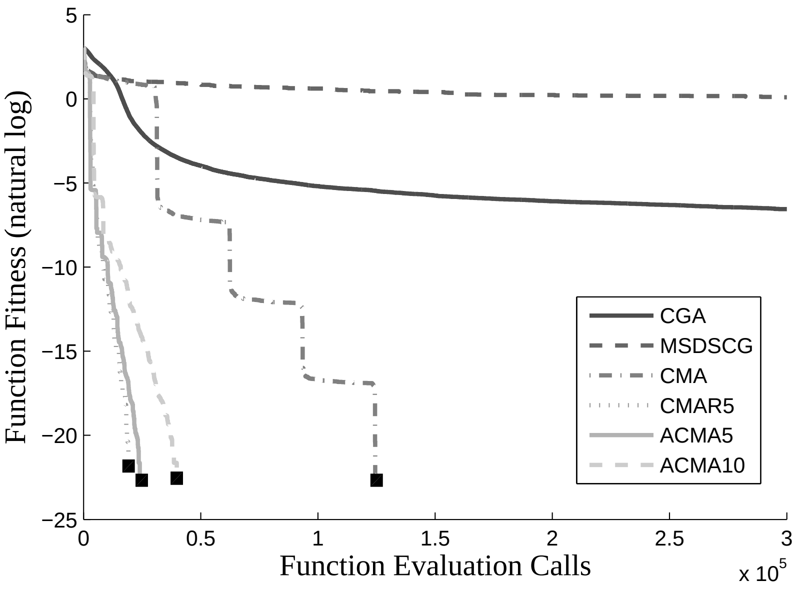

Figure 10: Search traces (average of 50 runs) for minimizing 30D Ackley function. (Square box indicates that the Global Optimum is found)

Here, we consider next the set of separable multimodal problems, i.e., functions imbue with low or no epistasis. Figures 10 - 12 present the search trends of the algorithms on the 30 dimensional Ackley, Rastrigin and Griewank problems which has increasing level of multimodality. The results indicate that the ACMAs and CMAs had converged to the global optimum while CGA fails to do so for the same termination condition, suggesting the DSCG local search synergizes well with the CGA to bring about search improvements. Since the DSCG local search operates on each dimension separately (see Figure 6), it functions efficiently on the separable problems. This

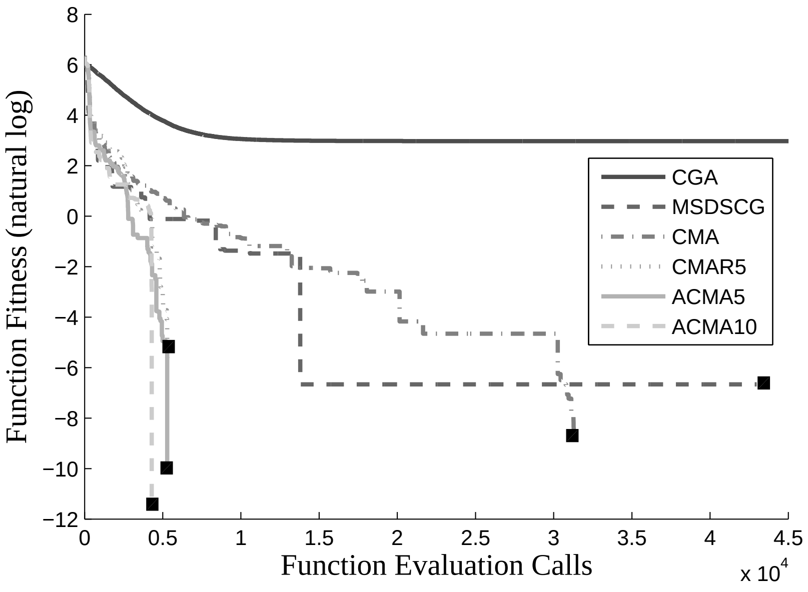

Figure 11: Search traces (average of 50 runs) for minimizing 30D Rastrigin function. (Square box indicates that the Global Optimum is found)

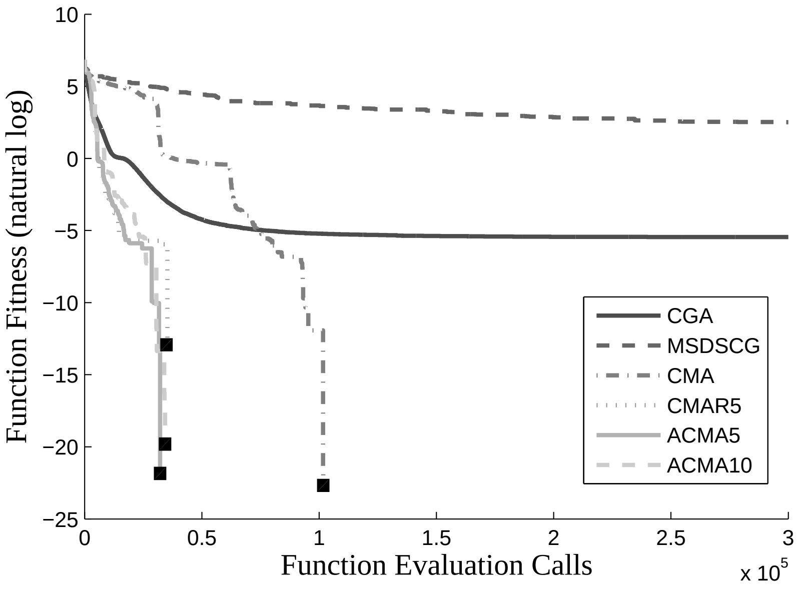

Figure 12: Search traces (average of 50 runs) for minimizing 30D Griewank function. (Square box indicates that the Global Optimum is found)

Table 5: Fitness error values and success rate (30 Dimensional Ackley)(note that the best performing algorithm is highlighted in bold typeface)

Algorithm	Fitness error values and success rate				
	Best	Worst	Mean	Std	Success
CGA	0.000819	0.002120	0.001418	0.000325	0%
MS-DSCG	0.064219	2.561123	1.098980	0.672054	0%
CMA	0.0 (124294)	0.0 (124477)	0.0 (124393)	0.0 (32)	100%
CMAR5	**0.0 (15462)**	**0.0 (20294)**	**0.0 (17699)**	**0.0 (1243)**	**100%**
ACMA5	0.0 (14775)	0.0 (24435)	0.0 (21027)	0.0 (1846)	100%
ACMA10	0.0 (26780)	0.0 (39853)	0.0 (33909)	0.0 (3074)	100%

Table 6: Fitness error values and success rate (30 Dimensional Rastrigin)(note that the best performing algorithm is highlighted in bold typeface)

Algorithm	Fitness error values and success rate				
	Best	Worst	Mean	Std	Success
CGA	9.949604	36.813473	19.560903	5.540093	0%
MS-DSCG	0.0 (239)	0.0 (34266)	0.0 (9645)	0.0 (8302)	100%
CMA	0.0 (343)	0.0 (31291)	0.0 (7973)	0.0 (7676)	100%
CMAR5	0.0 (350)	0.0 (5291)	0.0 (2713)	0.0 (1325)	100%
ACMA5	**0.0 (336)**	**0.0 (5291)**	**0.0 (2511)**	**0.0 (1111)**	**100%**
ACMA10	0.0 (332)	0.0 (4297)	0.0 (2695)	0.0 (1797)	100%

Table 7: Fitness error values and success rate (30 Dimensional Griewank)(note that the best performing algorithm is highlighted in bold typeface)

Algorithm	Fitness error values and success rate				
	Best	Worst	Mean	Std	Success
CGA	0.000020	0.019786	0.004263	0.005972	0%
MS-DSCG	5.792934	35.850984	12.249027	4.521426	0%
CMA	0.0 (62666)	0.0 (101777)	0.0 (84336)	0.0 (12715)	100%
CMAR5	**0.0 (8213)**	**0.0 (35290)**	**0.0 (14494)**	**0.0 (5274)**	**100%**
ACMA5	0.0 (7788)	0.0 (32194)	0.0 (15053)	0.0 (4962)	100%
ACMA10	0.0 (12057)	0.0 (34789)	0.0 (20027)	0.0 (6302)	100%

suggests why the MS-DSCG also managed to find the global optimum on Rastrigin problem on all the 50 runs.

Further, consistent with the results shown on the unimodal problems, CMAR5, ACMA5 and ACMA10 again attained superior search performances over the CMA. Clearly, this is attributed to the introduction of selective local search schemes into the CMA. To quantify the significance in improvements by the adaptive CMAs, the search statistics of the algorithms are also summarized in Tables 5-7. It is worth noting that besides a faster convergence speed, CMAR5, ACMA5 and ACMA10 also exhibit better search robustness as indicated by the lower standard deviation of the results obtained across the 50 independent runs.

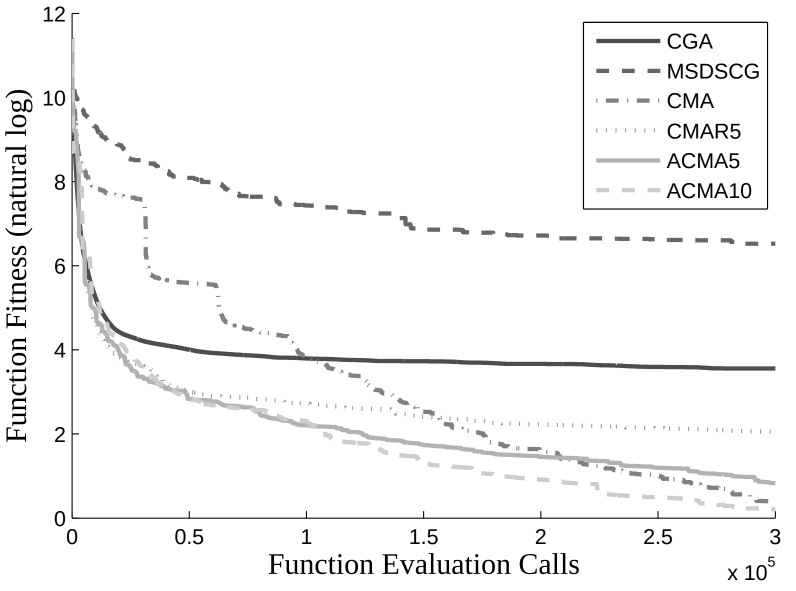

Figure 13: Search traces (average of 50 runs) for minimizing 30D Rosenbrock function.

Rosenbrock is a multi-modal function plague with strong epistasis. Although its fitness landscape is not as rugged as the other three previously considered benchmark

Table 8: Fitness error values and success rate (30 Dimensional Rosenbrock)(note that the best performing algorithm is highlighted in bold typeface)

Algorithm	Fitness error values and success rate				
	Best	Worst	Mean	Std	Success
CGA	0.100618	80.102006	34.995072	25.813214	0%
MS-DSCG	145.614786	2681.761437	681.762756	390.338469	0%
CMA	0.000089	15.750893	1.436206	2.725633	0%
CMAR5	0.007964	67.471566	7.762101	10.577379	0%
ACMA5	0.000072	16.184286	2.288657	3.677642	0%
ACMA10	**0.000466**	**15.473135**	**1.230378**	**2.738209**	**0%**

problems, the strong interactions between the genes is often deem to make it hard for many optimization approaches including GA or CGA. This suggests why the algorithms failed to locate the global optimum accurately, i.e., success hit rate of 0% on all the 50 independent runs (see Figure 13 and Table 8). Nevertheless, the results obtained highlighted that the inclusion of local search had contributed to improvements in the search performance of the CGA, i.e., CMA, CMAR5, ACMA5 and ACMA10 produces improved performance than the CGA. Further, it is observed that CMA and ACMA10 had performed better than CMAR5 and ACMA5, suggesting greater exploitation is beneficial on the Rosenbrock problem. On the other hand, ACMA5 is observed to be superior over the CMAR5 further suggests the latter facing difficulties with the strong epistasis in the problem. Hence a selection scheme with some uniformity such as the stratified scheme is deem to be more appropriate than a simple random selection.

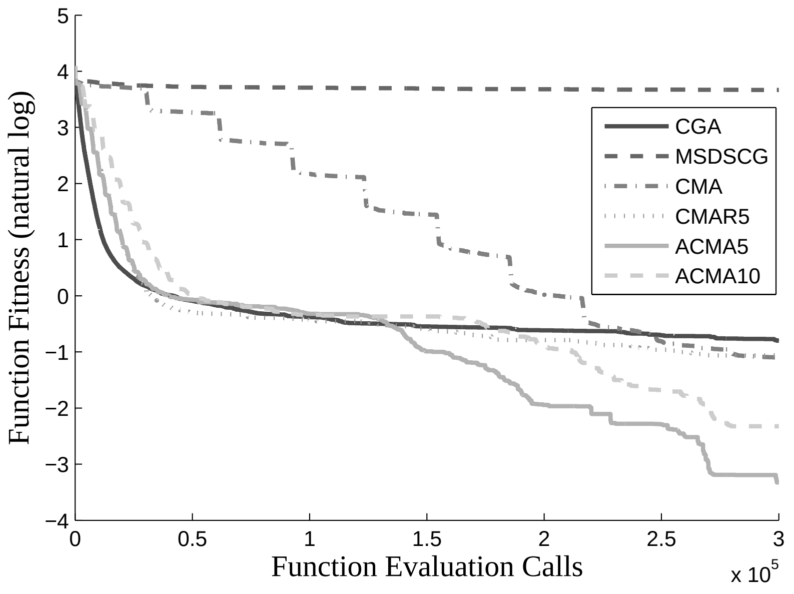

Figure 14: Search traces (average of 50 runs) for minimizing 30D Weierstrass function.

Table 9: Fitness error values and success rate (30 Dimensional Weierstrass)(note that the best performing algorithm is highlighted in bold typeface)

Algorithm	Fitness error values and success rate				
	Best	Worst	Mean	Std	Success
CGA	0.045592	3.530791	0.449901	0.759902	0%
MS-DSCG	35.269832	41.501266	39.144839	1.360618	0%
CMA	0.011849	2.412587	0.333060	0.601024	0%
CMAR5	0.002224	1.973621	0.345337	0.540117	0%
ACMA5	**0.002177**	**0.897336**	**0.035927**	**0.161454**	**0%**
ACMA10	0.001824	0.986958	0.097758	0.287558	0%

The extremely rugged surface of Weierstrass functions is generally a greater challenge for CMAs than the CGA. The significantly large number of basins reduce severely the efficacy of CMAs as the local search is easily trapped in the local optima and most of the computational efforts spent are generally wasted. This suggests why the CMA which conducts a local refinement on each individual in the CGA population fares poorly when compared to CMAR5, ACMA5 and ACMA10, for the same computational budget, see Table 9. It is worth noting that even though CMAR5 is able to achieve excellent search performance, the simple random selection scheme suffers from poor robustness as observed in the large standard deviations of the results on the Weierstrass problem (see Figure 14 and Table 9). In contrast, the ACMAs fare the best on the Weierstrass problem.

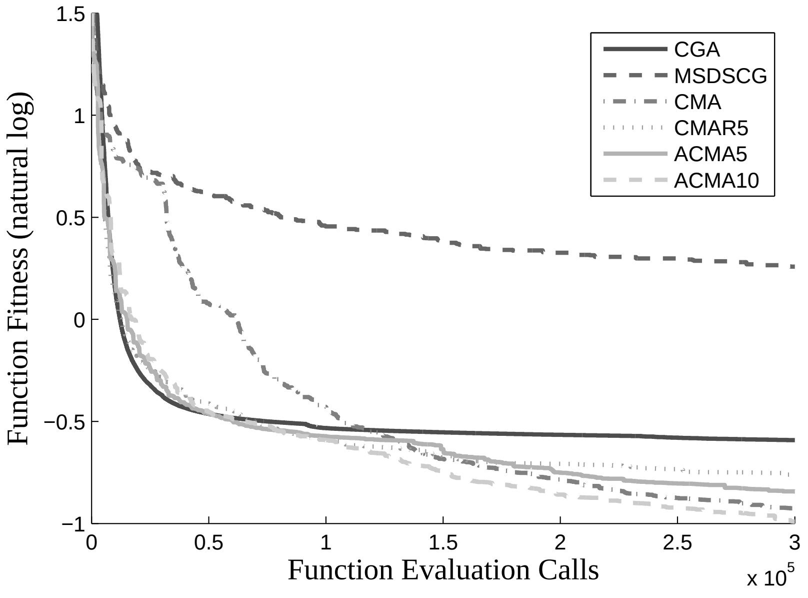

Figure 15: Search traces (average of 50 runs) for minimizing 30D Scaffer function.

On the Scaffer problem, it is observed in Figure 15 and Table 10 that the local search

Table 10: Fitness error values and success rate (30 Dimensional Scaffer)(note that the best performing algorithm is highlighted in bold typeface)

Algorithm	Fitness error values and success rate				
	Best	Worst	Mean	Std	Success
CGA	0.253642	1.509344	0.553448	0.297457	0%
MS-DSCG	0.671650	1.840983	1.294955	0.274535	0%
CMA	0.208198	0.783990	0.395896	0.150297	0%
CMAR5	0.077727	1.318264	0.467296	0.263380	0%
ACMA5	0.038864	0.936715	0.430312	0.223992	0%
ACMA10	**0.097159**	**1.276680**	**0.371344**	**0.213785**	**0%**

Table 11: Fitness error values and success rate (FMS)(note that the best performing algorithm is highlighted in bold typeface)

Algorithm	Fitness error values and success rate				
	Best	Worst	Mean	Std	Success
CGA	0.000063	25.470558	15.810940	7.312292	0%
MS-DSCG	0.049222	17.229564	6.981543	4.753076	0%
CMA	0.0 (71098)	14.778976	7.192541	5.606119	36%
CMAR5	0.0 (10670)	22.383644	13.097359	7.465197	20%
ACMA5	0.0 (4884)	21.483843	4.168509	6.189833	66%
ACMA10	**0.0 (6621)**	**17.382780**	**3.679700**	**5.595474**	**68%**

procedure, i.e., DSCG, does not bring about significant improvement to the search, since the CGA displays a search trend that is close to those of the CMAs. This suggests why MS-DSCG performs badly on the Scaffer problem. The search performances of the CMA, CMAR5, ACMA5 and ACMA10, on the other hand, do not differ significantly. Nevertheless, CMA and ACMA10 appear to fare slightly better than CMAR5 and ACMA5 in terms of solution quality and robustness.

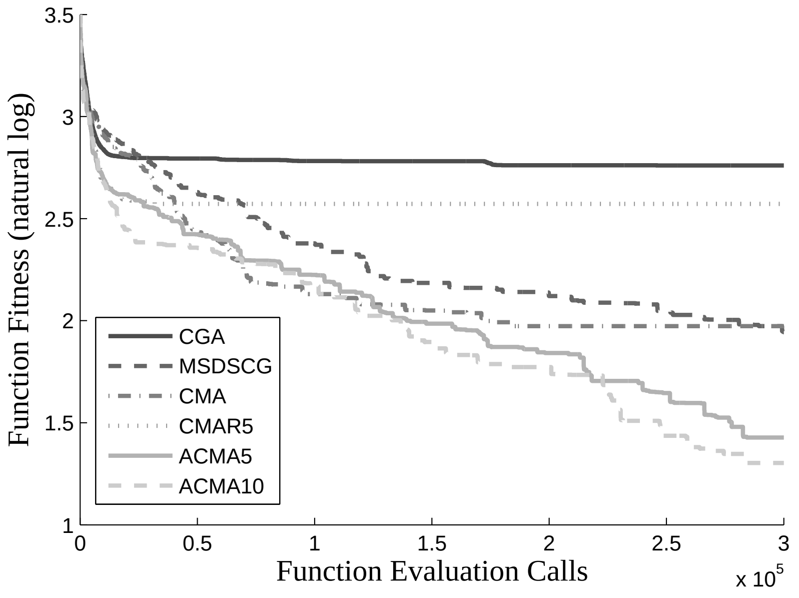

Figure 16: Search traces (average of 50 runs) for minimizing FMS function.

Lats but not least, we discuss the results of the search on the multimodal and highly epistatic FMS problem. Figure 16 and Table 11 indicate that ACMAs outperform the other counterparts significantly in terms of solution quality (best solution achievable), robustness and success rate. The consistently good performance of the ACMAs on this highly complex problem further emphasizes the importance of selection local search in CMAs.

In summary, it is now widely accepted that synergy between global and local search can bring about improved search performance. Nevertheless, our study has shown that a good balance through global exploration and local exploitation represents a crucial factor to achieving high and robust quality solutions efficiently. Exhaustive local search in CMA may lead to ineffective search due to the inefficient use of computational resources and premature fall in diversity during the search. On the other hand, random selection of individuals for local improvement may affect the robustness of the search. In such case, a diversity-based dynamic adaptive scheme such as the stratified scheme proposed would be more appropriate.

4.3 Cellular MAs with Comparison to Other Evolutionary and Memetic Approaches

In the following section, we present a comparison with other advanced evolutionary algorithms using the same set of benchmark problems studied in section 4.1. ACMA5 and ACMA10 is compared to the Orthogonal GA (OGA/Q) (Leung and Wang, 2001) and Hybrid Taguchi-Genetic Algorithm Tsai et al. (2004) on the 30D Sphere, Ackley, Rastrigin and Griewank. Figure 17 reports the number of function evaluation calls incurred by each algorithm in converging to a fitness error of 10^{-8} on the benchmark problems. The detailed statistical comparison between ACMA5 and the two algorithms are summarized in Tables 12 and 13. It is noted that ACMA5 and ACMA10 outperform the OGA/Q significantly at 99% confidence level on all the benchmark problems. At the same time, ACMA5 and ACMA10 also outperform HTGA on all the problems considered significantly, except the Ackley function.

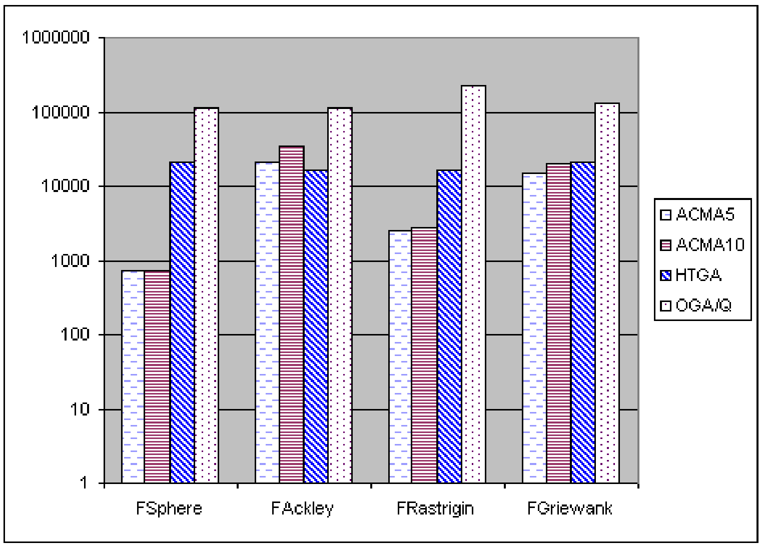

Figure 17: Number of function evaluations used by different algorithms in solving the 30D benchmark functions.

Table 12: Statistic test on comparing ACMA5 vs. OGA/Q (Null hypothesis: ACMA5 and OGA/Q incur equal number of evaluations vs. alternative hypothesis: ACMA5 and OGA/Q incur different number of evaluations to reach to the global optimum)

Problems	ACMA5	OGA/Q	t-value	p-value	Conclusion
Sphere	742 ± 603	112559	-1311.2	$5.00e^{-113}$	ACMA5 is better
Ackley	21027 ± 1846	112421	-350.1	$6.25e^{-85}$	ACMA5 is better
Rastrigin	2511 ± 1111	224710	-1414.2	$1.235e^{-114}$	ACMA5 is better
Griewank	15053 ± 4962	134000	-169.1	$1.65e^{-69}$	ACMA5 is better

Table 13: Statistic test on comparing ACMA5 vs. HTGA (Null hypothesis: ACMA5 and HTGA incur equal number of evaluations vs. alternative hypothesis: ACMA5 and HTGA incur different number of evaluations to reach to the global optimum)

Problems	ACMA5	HTGA	t-value	p-value	Conclusion
Sphere	742 ± 603	20844	-235.7	$1.61e^{-76}$	ACMA5 is better
Ackley	21027 ± 1846	16632	16.8	$2.60e^{-22}$	HTGA is better
Rastrigin	2511 ± 1111	16267	-87.6	$1.67e^{-55}$	ACMA5 is better
Griewank	15053 ± 4962	20999	-8.5	$3.64e^{-11}$	ACMA5 is better

5 Conclusion

Previous studies have shown that the Cellular GA possesses better population diversity and exploration capacity than the standard GA (Kohlmorgen et al., 1999; Alba and Troya, 2002). The impact of reducing the effect of premature convergence however raises the issue of slow convergence rate in CGAs. We have identified this as a core challenge for CGA especially when used in solving optimization problems that are computationally demanding. Since it is common knowledge that local search methods can locate local optima effectively and efficiently, it can potentially complement the CGA as a means of increasing its exploitation ability.

In this paper, we have analyzed the performance of the cellular GA against the canonical cellular MA on a series of benchmark problems representing classes of unimodal, multimodal test functions having different level of epistasis. Empirical results on the test functions show that ACMA locates global optimum more often than the CGA; generating better success hit rates and giving the best optimum solution with lower computational effort, i.e., a lower minimum error threshold and a smaller number of function evaluations. However, due to the aggressive exploitative nature of the cellular MA, it is more susceptible to getting stuck in local optima than the cellular GA.

To achieve a desirable level of performance in a search algorithm, neither exploitation nor exploration should dominate. We have shown that adaptive cellular MAs (ACMAs) with selective local search mechanism can balance the degree of exploitation and exploration in the search under limited computational budget. Empirical study shows that the proposed ACMA significantly outperforms the Cellular GA, canonical Cellular MA and the Cellular MA with random selection on all the benchmark problems tested, in terms of solution quality, search efficiency and robustness.

Acknowledgement

This work has been funded in part under the A*STAR SERC Grant No. 052 015 0024 administered through the National Grid Office. NK acknowledges the EPSRC grants EP/E017215/1, EP/D021847/1 and EP/D061571/1. The authors would also like to thank W. Tjiawi for his help on the present work.

References

Alba, E. and Dorronsoro, B. (2005). The exploration/exploitation tradeoff in dynamic cellular evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 9(2):126–142.

Alba, E., Dorronsoro, B., and Alfonso, H. (2005). Cellular Memetic Algorithms. *Journal of Computer Science & Technology*.

Alba, E., Giacobini, M., Tomassini, M., and Romero, S. (2002). Comparing synchronous and asynchronous cellular genetic algorithms. *Parallel Problem Solving from Nature-PPSN VII*, 2439:601–610.

Alba, E. and Troya, J. M. (2002). Improving flexibility and efficiency by adding parallelism to genetic algorithms. *Statistics and Computing*, 12(2):91–114.

Baluja, S. (1993). Structure and performance of fine-grain parallelism in genetic search. *Proceeding of the Fifth International Conference on Genetic Algorithms*, pages 155–162.

Bradwell, R., Brown, K., Cantu-Paz, E., and Punch, B. (1999). Parallel asynchronous memetic algorithms. *Evolutionary Computation*, pages 157–159.

Burke, E., Cowling, P., De Causmaecker, P., and Berghe, G. (2001). A Memetic Approach to the Nurse Rostering Problem. *Applied Intelligence*, 15(3):199–214.

Burke, E., Gustafson, S., Kendall, G., and Krasnogor, N. (2002). Advanced population diversity measures in genetic programming. In Merelo-Guervs, J., Adamidis, P., Beyer, H., Fernndez-Villacaas, J., and Schwefel, H., editors, *7th International Conference Parallel Problem Solving from Nature*, volume 2439 of *Springer Lecture Notes in Computer Science*, pages 341–350, Granada, Spain. PPSN, Springer Berlin / Heidelberg. ISBN 3-540-44139-5.

Burke, E., Gustafson, S., Kendall, G., and Krasnogor, N. (2003). Is increased diversity beneficial in genetic programming: An analysis of the effects on fitness. In *IEEE Congress on Evolutionary Computation*, pages 1398–1405. CEC, IEEE.

Deb, K. and Beyer, H. (2001). Self-Adaptive Genetic Algorithms with Simulated Binary Crossover. *Evolutionary Computation*, 9(2):197–221.

Digalakis, J. G. and Margaritis, K. G. (2000). A performance comparison of parallel genetic and memetic algorithms using MPI. *IntRep03, University of Macedonia, Parallel Distributed Processing Laboratory, Thessaloniki, Greece*.

Digalakis, J. G. and Margaritis, K. G. (2001). On benchmarking functions for genetic algorithms. *Intern. J. Computer Math.*, 77(4):481–506.

Folino, G., Pizzuti, C., and Spezzano, G. (1998). Combining cellular genetic algorithms and local search for solving satisfiability problems. *Tenth IEEE International Conference on Tools with Artificial Intelligence, 1998.*, pages 192–198.

Goldberg, D. and Voessner, S. (1999). Optimizing Global-Local Search Hybrids. *Urbana*, 51:61801.

Hart, W., Krasnogor, N., and Smith, J. (2004a). Editorial introduction, special issue on memetic algorithms. *Evolutionary Computation*, 12(3):v–vi.

Hart, W., Krasnogor, N., and Smith, J. (2004b). *Recent advances in memetic algorithms*, volume 166 of *Studies in Fuzzyness and Soft Computing*. Springer Berlin Heidelberg New York. ISBN 3-540-22904-3.

Hart, W. E. (1994). *Adaptive Global Optimization with Local Search*. PhD thesis, University of California, San Diego.

Hasan, S., Sarker, R., Essam, D., and Cornforth, D. (2009). Memetic algorithms for solving job-shop scheduling problems. *Memetic Computing*, 1(1):69–83.

Houck, C., Joines, J., Kay, M., and Wilson, J. (1997). Empirical Investigation of the Benefits of Partial Lamarckianism. *Evolutionary Computation*, 5(1):31–60.

Ishibuchi, H., Yoshida, T., and Murata, T. (2003). Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. *IEEE Transactions on Evolutionary Computation*, 7(2):204–223.

Kohlmorgen, U., Schmeck, H., and Haase, K. (1999). Experiences with fine-grained parallel genetic algorithms. *Annals of Operations Research*, 90:203–219.

Krasnogor, N. and Smith, J. (2000). A memetic algorithm with self-adaptive local search: TSP as a case study. *Proceeding of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 987–994.

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

Krasnogor, N. and Smith, J. (2008). Memetic algorithms: The polynomial local search complexity theory perspective. *Journal of Mathematical Modelling and Algorithms*, 7(1):3–24.

Leung, Y. W. and Wang, Y. (2001). An orthogonal genetic algorithm with quantization for globalnumerical optimization. *IEEE Transactions on Evolutionary Computation*, 5(1):41–53.

Lim, M. H. and Xu, Y. L. (2005). Application of hybrid genetic algorithm in supply chain management. *Special issue on Multi-Objective Evolution: Theory and Applications, International Journal of Computers, Systems, and Signals*, 6(1).

Martin, W. N., Lienig, J., and Cohoon, J. P. (1997). Island (migration) models: evolutionary algorithms based on punctuated equilibria. *Handbook of Evolutionary Computation*, 6:3.

Muhlenbein, H., Schomisch, M., and Born, J. (1991). The Parallel Genetic Algorithm as a Function Optimizer. *Proceeding of the Fourth International Conference on Genetic Algorithms*, pages 271–78.

Ong, Y., Lim, M., Neri, F., and Ishibuchi, H. (2008). Special issue on emerging trends in soft computing: memetic algorithms. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, pages 739–740.

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

Ong, Y. S., Krasnogor, N., and Ishibuchi, H. (2007). Special Issue on Memetic Algorithms. *IEEE Transactions on Systems, Man and Cybernetics - Part B*, 37(1):2–5.

Ong, Y. S., Lim, M. H., Zhu, N., and Wong, K. W. (2006). Classification of Adaptive Memetic Algorithms: A Comparative Study. *IEEE Transactions on Systems, Man and Cybernetics – Part B.*, 36(1):141.

Pettey, C. (1997). *Diffusion (cellular) models. Handbook of Evolutionary Computation*. IOP Publishing.

Spiessens, P. and Manderick, B. (1991). A massively parallel genetic algorithm: Implementation and rst analysis. *Proceeding of the Fourth International Conference on Genetic Algorithms, San Diego, CA*, pages 279–287.

Tang, J., Lim, M. H., and Ong, Y. S. (Dec 2006). Parallel Memetic Algorithm with Selective Local Search for Large Scale Quadratic Assignment. *Intl. Journal of Innovative Computing, Information and Control*, 2(6):1399–1416.

Tang, J., Lim, M. H., and Ong, Y. S. (July 2007). Diversity-Adaptive Parallel Memetic Algorithm for Solving Large Scale Combinatorial Optimization Problems. *Soft Computing Journal*, 11(9):873–888.

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

Tsutsui, S. and Fujimoto, Y. (1993). Forking Genetic Algorithm with Blocking and Shrinking Modes (fGA). *Proceeding of the 5th International Conference on Genetic Algorithms table of contents*, pages 206–215.

Yu, X., Tang, K., Chen, T., and Yao, X. (2009). Empirical analysis of evolutionary algorithms with immigrants schemes for dynamic optimization. *Memetic Computing*, 1(1):3–24.