Differential Evolution with Self Adaptive Local Search

Nasimul Noman
noman@iba.t.u-tokyo.ac.jp

Danushka Bollegala
danushka@iba.t.u-tokyo.ac.jp

Hitoshi Iba
iba@iba.t.u-tokyo.ac.jp

Iba Laboratory, Graduate School of Engineering
University of Tokyo
Tokyo, Japan

ABSTRACT

The performance of a memetic algorithm (MA) largely depends on the synergy between its global and local search counterparts. The amount of global exploration and local exploitation to be carried out, for optimal performance, varies with problem type. Therefore, an algorithm should intelligently allocate its computational efforts between genetic search and local search. In this work we propose an adaptive local search method that adjusts the effort for local tuning of individuals, taking feedback from the search. We implemented an MA hybridizing this adaptive local search method with differential evolution algorithm. Experimenting with a standard benchmark suite it was found that the proposed MA can utilize its global and local search components adaptively. The proposed algorithm also exhibited very competitive performance with other existing algorithms.

Categories and Subject Descriptors

G.1.6 [**Numerical Analysis**]: Optimization—*Unconstrained optimization, Stochastic programming, Global optimization*

General Terms

Algorithms

Keywords

Differential Evolution, Memetic Algorithm, Adaptive MA, Adaptive Local Search

1. INTRODUCTION

The term "memetic algorithm" (MA) was coined by Moscato [11], that refers to an evolutionary algorithm (EA) coupled with local search (LS) within the evolutionary framework. Hybridization of EAs with LSs is also known as hybrid genetic algorithms [7], genetic local searches [3], Lamarckian GAs [10], Baldwinian GAs [21] etc. MAs are recent development in the field of EA and are acknowledged as very effective search meta-heuristics.

GECCO'11, July 12–16, 2011, Dublin, Ireland.

EAs are population based stochastic search algorithms competent for locating (pseudo) optimum solutions through exploration and exploitation of the search space. However, EAs are often criticized for not being suitable for fine tuning as they might take too much time to locate the exact solution [12, 14]. On the other hand LS algorithms get easily trapped in local minima, though they can locate the local optimum very quickly. Therefore, hybridization of EA with LS forms a very effective search framework by taking advantage of both paradigms of global search (GS) and local search (LS) [16].

Although MAs can serve as effective tools for global optimization, the success depends on the balance between the exploration, by the global search component, and the exploitation, by the local search process [5]. And the tradeoff between GS and LS is very critical for the performance of MAs, because they interact with the search space very differently. Failing to strike a balance between GS and LS not only wastes computational effort (fitness evaluations) but also results in premature convergence.

Previous studies have shown that the performance of MAs is dependent on LS operator [6, 15], frequency of LS [5], choice of individuals for LS [4] and intensity of LS [14, 8]. The influence of these design issues on the performance of MAs has motivated the design of self-adaptive MAs.

Ishibuchi *et al.* tried to adjust the balance between GS and LS by using three parameters: local search probability (P_{LS}), intensity of LS (k) and local search application interval (T) in their MOGLS algorithm [5] and showed that improper choice of these parameters may drive their algorithm to perform very poor. Ong and Keane [15] proposed an adaptive choice of LS operators to ensure robustness in MAs. In their method they used a pool of LSs and based on the online performance of different LSs in the pool, the selection of LS is biased. Bambha *et al.* [1] introduced a simulated heating framework that systematically incorporates parameterized LS into the framework of GS.

Lozano *et al.* [8] used a crossover based LS (XLS) method and an adaptation mechanism to decide which individual should undergo LS process. In this way, they attempted to adjust the global/local search ratio. Capino *et al.* [2] proposed an adaptive control to balance the needs of exploration and exploitation dynamically using a dynamic parameter setting and adaptive use of two LS with different structures. Smith proposed a framework for coevolving memes encoding definitions used in an MA for creating a robust scalable optimization platform [18]. Noman and Iba [14] tried to adjust the intensity of LS length by using a hill-climbing method.

Within the framework of cellular genetic algorithm (cGA), Nguyen *et. al* [13], proposed a stratified adaptation method for selecting suitable individuals to undergo LS operation. Nguyen *et. al* [12] proposed a probabilistic framework for MA for deciding online whether evolution or individual learning should be favored for accelerating the algorithm. They also derived a theoretical bound for LS intensity that was incorporated in their framework. Molina *et al.* [9] proposed the concept of local search chain in order to use intense continuous LS operators effectively in MAs.

All these recent work highlight the importance of adaptation in MA. Following this recent trend in MA, in this work we propose an adaptive local search for dynamically balancing the degree of GS and LS. Based on online search performance, the adaptive LS scheme selects the number of individuals that should undergo LS operation and also adjusts the maximum intensity of the local tuning of each selected individual. We hybridized this adaptive LS scheme with differential evolution (DE), a prominent real-valued EA, in the general template of MA. We evaluated the proposed MA with a standard benchmark suite consisting of ten test problems with different characteristics. The experimental results show that the proposed MA can adaptively favor its evolutionary and local improvement counterparts and thereby balance the exploration and exploitation capacity of the algorithm. Comparing with other MAs the performance of the proposed algorithm was found to be very competitive.

2. THE CANONICAL MA AND DE

This section reviews the framework of canonical memetic algorithms (MAs) and then overviews the differential evolution (DE) algorithm briefly. This recap will help to formalize our proposed adaptive MA which will be presented next.

2.1 Memetic Algorithm

Apart from a few disagreements, most people recognize the term "memetic algortihm" as some sort of hybridization between EAs and LSs. Generally, the LS is incorporated within the template of EA such that both components work cooperatively to explore the search space for global optimum [11, 18]. A very general framework of MA is shown in Algorithm 1.

Algorithm 1 MA

1: P = initialize population randomly
2: P = ***applyLS***(P)
3: **while** termination criteria not satisfied **do**
4: Select parent individuals from P
5: Apply crossover and mutation to generate offspring C from parents
6: P = Select (P,C)
7: P = ***applyLS***(P)
8: **end while**

The template of Algorithm 1 gives the impression of MA that incorporates the LS within its evolutionary loop. However, this simple "applyLS()" operation actually is an oversimplified representation of integrating LS in the evolutionary cycle. It hides all the design issues: when to apply, whom to apply, how long to apply, what to apply etc., that are crucial for the performance of the algorithm. Nevertheless, from Algorithm 1, it is evident that an MA is actually an interplay between EA and LS whose success depends on useful cooperation and beneficial competition between them.

2.2 Differential Evolution

Differential Evolution (DE) is a very powerful and reliable optimizer for continuous search spaces [19, 20]. The algorithm has many attractive characteristics compared to other EAs, such as simple and easy-to-understand structure, few controlling parameters, superior convergence characteristics and robust performance. DE has proven to be very effective in solving non-linear, non-differentiable, non-convex and multi-modal optimization problems. Due to its robust performance, DE has found many applications in real world problems.

DE works with a population of N-dimensional vectors $\boldsymbol{x}_G^i$, $i = 1, 2, \cdots, P$. Each vector $\boldsymbol{x}_G^i$ represents a solution in the N dimensional search space. P denotes the number of individuals in the population and G denotes the current generation. Initial population is created by randomly creating the vectors in appropriate search ranges. Then each individual is evaluated to determine its fitness.

DE does not apply selection pressure for selecting parents. Rather, in each generation, every individual $\boldsymbol{x}_G^i$, once becomes the principal parent to breed it own offspring mating with other parents. These auxiliary parents are chosen randomly. Formally, for every principal parent $\boldsymbol{x}_G^i$, $i = 1, 2, \cdots, P$, three other auxiliary parents $\boldsymbol{x}_G^{r1}$, $\boldsymbol{x}_G^{r2}$, $\boldsymbol{x}_G^{r3}$ are selected randomly such that $r1, r2, r3 \in \{1, 2, \cdots, P\}$ and $i \neq r1 \neq r2 \neq r3$. Then these three auxiliary parents take part in *differential mutation* operation to create a trial individual $\boldsymbol{x}_G^{trial}$ as follows:

$$\boldsymbol{x}_G^{trial} = \boldsymbol{x}_G^{r1} + F(\boldsymbol{x}_G^{r2} - \boldsymbol{x}_G^{r3}) \tag{1}$$

where F is the amplification factor, a positive real number typically less than 1 [17].

Then the trial vector, $\boldsymbol{x}_G^{trial}$, with the principal parent $\boldsymbol{x}_G^i$, participates in a crossover operation to generate the offspring $\boldsymbol{x}_G^{child}$. Two different crossover operations, binomial and exponential, were suggested for DE in its original proposal [20], though in principle any real-valued crossover operation is possible. Both of the above mentioned crossover operations are administrated by another parameter called crossover probability, $C_r \in [0, 1.0]$. In binomial crossover, genes of $\boldsymbol{x}_G^{child}$ are inherited from either $\boldsymbol{x}_G^{trial}$ or $\boldsymbol{x}_G^i$ with probability C_r. In exponential crossover, C_r determines how many consecutive genes of the trial vector, $\boldsymbol{x}_G^{trial}$, on average are copied to the offspring $\boldsymbol{x}_G^{child}$. The exponential crossover was used in our implementation of DE in this work.

The selection scheme used in DE is also known as *parent-child competition*. As the name suggests, in order to select the survivor DE employs a deterministic binary knock-out competition between each individual $\boldsymbol{x}_G^i$ and its offspring $\boldsymbol{x}_G^{child}$ as follows:

$$\boldsymbol{x}_{G+1}^i = \begin{cases} \boldsymbol{x}_G^{child} & \text{if } f(\boldsymbol{x}_G^{child}) \text{ is better than } f(\boldsymbol{x}_G^i) \\ \boldsymbol{x}_G^i & \text{otherwise} \end{cases} \tag{2}$$

Besides, several other variants of DE, with different learning strategies, exist about which can be learnt from [17].

3. DE WITH ADAPTIVE LS

A new adaptive LS scheme to balance the exploration and exploitation ratio in an MA, is presented in this section.

Later we describe how this LS is incorporated within DE to implement an adaptive MA. The success of an MA lies in the synergy between its GS and LS components so that the exploration and exploitation capabilities of the algorithm can be balanced. However, the performance of GS and/or LS heavily depends on the problem type. Even their performances may vary at different stages of the search. Therefore, it should be profitable if the GS or LS components' CPU share can be adjusted according to their performance.

Driven by the above notion, here we propose an adaptive LS scheme that controls the maximum exploitation allowed for LS based on its online performance. We keep track of the performance of both GS (GS_{perf}) and LS (LS_{perf}) in recent generations. If LS is performing better than GS then we increase our preference to exploitation and if LS is performing poor then we reduce our favor for LS. We regulate our preference to LS by adjusting the number of individuals that will receive LS (N_{LS}) and the maximum LS intensity (I_{LS}). The adaptation scheme is shown in Algorithm 2. It should be noted that by adjusting N_{LS} and I_{LS} we can reciprocally administer our preference to LS or GS.

Algorithm 2 AdaptLS

```
Calculate LS_perf and GS_perf
if LS_perf > GS_perf then
    Increase I_LS by q%
    Increase N_LS by q%
    if N_LS > 0.5 × |P| then
        Set N_LS = 0.5 × |P|
    end if
else
    Decrease I_LS by q%
    Decrease N_LS by q%
    if N_LS < 1 then
        Set N_LS = 1
    end if
end if
```

In order to implement this adaptive LS algorithm we need to measure the performance of the search components. To estimate the performance of a search algorithm we keep track of the total number of search operations (fitness evaluations) applied and the total fitness improvement achieved in these search operations. Each time a child has better fitness than its parent, the absolute value of their fitness difference is added to fitness improvement. Then the performance of a search algorithm (S) can be defined as follows:

$$S_{perf} = \frac{\text{Total fitness improvement achieved by } S}{\text{Total fitness evaluation used in } S} \quad (3)$$

Another issue involved in the design of this adaptive LS operation is the choice of individuals to receive LS. The most favorable candidates for LS operation are the individuals with higher fitness as they are perhaps the individuals closer to the basin of attraction [8, 14]. On the other hand too much exploitation of elite individuals may result into premature convergence. In order to balance between these two contradictory criteria, we first include the best individual and then other ($N_{LS} - 1$) random individuals from the current generation in our LS pool.

The final decision is about choosing the LS operator. In some recent work, it has been shown that the Davies, Swann, and Campey method with Gram-Schmidt orthogonalization (DSCG) is a very good choice as an LS operator in continuous domain [15]. Moreover, some very recent MAs have been implemented using DSCG [13]. Therefore, in this work we used DSCG as our LS operator which also gives us the opportunity to compare our algorithm with these MAs more fairly.

Algorithm 3 DEaLS

```
P_G = initialize population randomly
Initialize N_LS and I_LS
P_G = applyLS(P_G)
while termination criteria not satisfied do
    for each individual x^i in P_G do
        Select auxiliary parents
        Create offspring x^c using mutation and crossover
        P_{G+1} = P_{G+1} ∪ Best(x^c, x^i)
    end for
    Set G = G + 1
    P_G = applyLS(P_G)
    if G mod G_ADJ == 0 then
        Call AdaptLS
    end if
end while
```

The adaptive LS scheme was integrated within the framework of DE to implement an adaptive MA. The newly developed MA is called DEaLS which is shown in Algorithm 3. DEaLS works with DE's genetic operators and DSCG is used as LS operator. The algorithm also keeps track of fitness evaluation and fitness improvement achieved by its GS and LS operators. And after each G_{ADJ} generations, it adapts its degree of exploration and exploitation based on the performance track of GS/LS in last G_{ADJ} generations.

4. EXPERIMENTAL RESULTS

In this section we present the numerical study done on the newly proposed DEaLS algorithm to evaluate its performance. In order to justify the effectiveness of the proposed adaptive LS scheme we compared it with another static LS scheme where the N_{LS} and I_{LS} parameters were initialized with the same values as in DEaLS but were kept constant throughout the search. And all the N_{LS} individuals, receiving LS, were chosen randomly. We call this algorithm DE with fixed LS (DEfLS). We also compared the newly proposed algorithm with the original DE algorithm under the same parameter settings. Additionally, to justify the competitiveness of the proposed algorithm we compared it with the multi start DSCG (MS-DSCG) and two other adaptive MAs (ACMA5 and ACMA10) proposed in [13]. In all of these MAs, DSCG was used as the LS operator. And the experimental results of MS-DSCG, ACMA5 and ACMA10, presented here, are collected from [13].

4.1 Test Suite and Performance Measure

Before presenting the results of numerical study we describe the benchmark suite and the evaluation criteria used to compare different algorithms. The test suite consists of ten benchmark functions commonly used for algorithms' performance evaluation. Three of these benchmark functions are unimodal problems: Sphere (f_{sph}), Elliptic (f_{ell}) and Schwefel1.2 (f_{sch}); six are multimodal problems: Ackley (f_{ack}), Rastrigin (f_{ras}), Griewank (f_{grw}), Rosenbrock

Table 1: Performance comparison for f_{sph} ($N = 30$) Error Value (Fitness Evaluation)

Algorithm	Best	Worst	Average	SD	Success
MS-DSCG †	0.0 (344)	0.0 (6070)	0.0 (1365)	0.0 (1111)	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%
DE	1.38E-37 (91023)	1.26E-36 (95125)	4.76E-37 (93189.4)	2.18E-37 (909.17)	100%
DEfLS	7.17E-39 (343)	2.10E-38 (376)	1.35E-38 (361.0)	2.56E-39 (8.07)	100%
DEaLS	**3.08E-67 (334)**	**9.87E-66 (372)**	**1.89E-66 (356.5)**	**1.54E-66 (8.48)**	**100%**

† In [13] error values is assumed to be 0.0 if $< 10^{-8}$

Table 2: Performance comparison for f_{ell} ($N = 30$) Error Value (Fitness Evaluation)

Algorithm	Best	Worst	Average	SD	Success
MS-DSCG	0.0 (344)	0.0 (8191)	0.0 (1688)	0.0 (1686)	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%
DE	4.41E-34 (116162)	4.93E-33 (121413)	1.69E-33 (118656.8)	8.73E-34 (1100.23)	100%
DEfLS	5.38E-35 (346)	2.05E-34 (376)	1.19E-34 (360.6)	3.60E-35 (6.94)	100%
DEaLS	**7.08E-63 (335)**	**2.79E-61 (375)**	**5.92E-62 (358.1)**	**5.14E-62 (9.08)**	**100%**

† In [13] error values is assumed to be 0.0 if $< 10^{-8}$

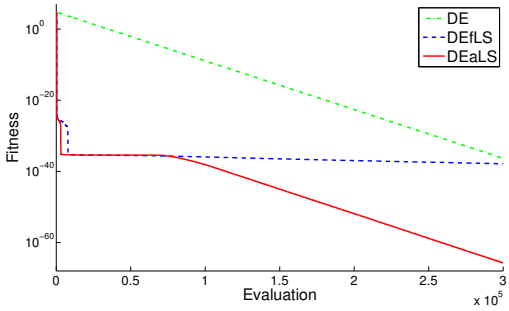

Figure 1: Convergence curve for f_{sph} function.

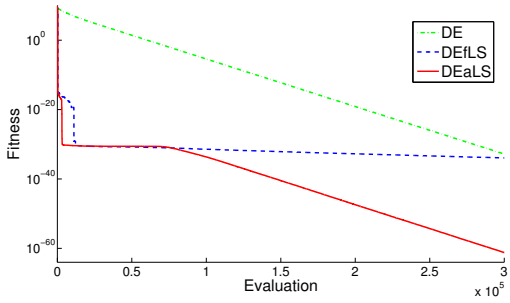

Figure 2: Convergence curve for f_{ell} function.

(f_{ros}), Weierstrass (f_{wrs}) and Schaffer (f_{scf}); and the last one is a real world problem, the frequency modulation sound (f_{fms}). All these problems were studied at 30 dimension except the real world problem which is a six-dimensional problem. This is the same test suite used in [13] and the definition of these benchmark functions can be found in [13] and in many other places in literature.

Each of these problems was solved in 50 independent trial runs starting from different random initial solutions. In each of these trials, we allowed an algorithm $N * 10,000$ fitness evaluations at maximum to solve a problem. We recorded the best error value (fitness value) achieved at the end of the trial and also kept track of the number of fitness evaluation required to achieve the error value $< 10^{-8}$ if the algorithm could achieve that. Then the minimum, maximum, average, and standard deviations of these error values and the required fitness evaluations were used for comparison. The notation used in comparison tables is $xxx(yyy)$ which indicates that an algorithm could reach this xxx error value at the end of the search and it took yyy fitness evaluations to reach an error value $< 10^{-8}$. We also compared in terms of success rate which is the rate of reaching the error value $< 10^{-8}$.

4.2 Results and Analysis

The experimental setup for DE, DEfLS and DEaLS was as follows: population size $P = 100$, amplification factor $F = 0.5$, crossover probability $C_r = 0.9$, initial values for $N_{LS} = 5$, $I_{LS} = 300$, $q = 10$ and $G_{ADJ} = 10$. The function-wise comparative results are presented in Table 1 to 10 and Fig. 1 to 10. The best results are marked in boldface.

For f_{sph} and f_{ell} functions, which are unimodal, symmetric functions without epistasis, the MS-DSCG could locate the global optimum very quickly as shown in Table 1 and 2 respectively. When this LS was hybridized with EAs the resulting MAs reasonably performed better than the LS alone. But the convergence rate of canonical DE was slow as shown in Fig. 1 and 2. In terms of fitness evaluations (FE) the performance of DEaLS and DEfLS was similar. This is because both algorithms reached the error value $< 10^{-8}$ before any adaptation could take place in DEaLS. But because of the inclusion of elite individual in adaptive LS scheme, DEaLS performed little better than DEfLS. However, in terms of final fitness value the performance of DEaLS was far better than DEfLS as well as than other algorithms.

The other unimodal function, f_{sch}, is asymmetric and partially separable. Therefore, this problem was difficult for MS-DSCG to solve. It was also difficult for DE and after hybridizing the fixed LS with DE the performance (of DEfLS) deteriorated. Only the DEaLS scheme could solve the problem in every trial run and the benefit of adaptation in LS is understandable from both Table 3 and Fig. 3. It seems from the graph of Fig. 3 that using adaptation DEaLS performed at least as good as DE in the initial stage of search

Table 3: Performance comparison for f_{sch} ($N = 30$) Error Value (Fitness Evaluation)

Algorithm	Best	Worst	Average	SD	Success
MS-DSCG	25017.911043	46222.875190	36613.290116	4981.768773	0%
ACMA5	0.000694	0.084662	0.013748	0.016618	0%
ACMA10	0.017464	3.542390	0.509196	0.548611	0%
DE	1.89E-04	1.39E-03	4.32E-04	2.01E-04	0%
DEfLS	5.43E+01	8.29E+02	3.06E+02	1.83E+02	0%
DEaLS	**9.74E-24 (74734)**	**1.83E-13 (118506)**	**3.66E-15 (85849.8)**	**2.59E-14 (7216.52)**	**100%**

Table 4: Performance comparison for f_{ack} ($N = 30$) Error Value (Fitness Evaluation)

Algorithm	Best	Worst	Average	SD	Success
MS-DSCG	0.064219	2.561123	1.098980	0.672054	0%
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%
DE	2.66E-15 (140402)	6.22E-15 (147350)	4.23E-15 (144219.2)	1.78E-15 (1383.24)	100%
DEfLS	0.00E+00 (20956)	0.00E+00 (58309)	0.00E+00 (43044.5)	0.00E+00 (9046.05)	100%
DEaLS	**0.00E+00 (6648)**	**0.00E+00 (10569)**	**0.00E+00 (7601.8)**	**0.00E+00 (890.91)**	**100%**

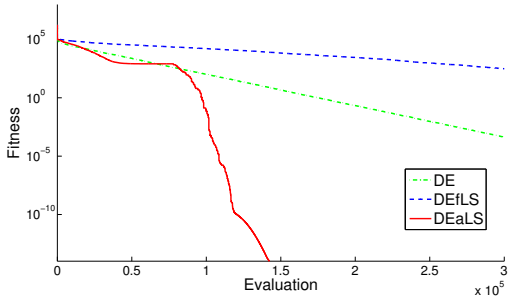

Figure 3: Convergence curve for f_{sch} function.

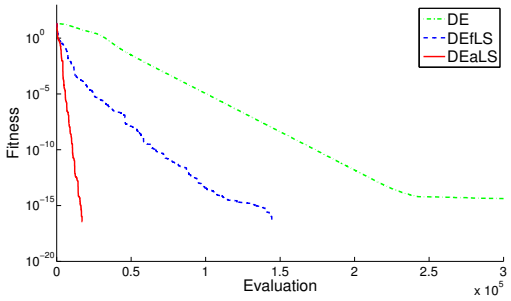

Figure 4: Convergence curve for f_{ack} function.

and then the synergic effect of LS and GS could drive the search towards global optimum.

Among the multimodal problems the f_{ack} is non-separable and symmetric. Hence, MS-DSCG could not solve it alone but when DSCG was hybridized with some GS, the MA could locate the optimum in every trial. Although DE itself could locate the global optimum, the convergence speed was relatively very slow. Table 4 and Fig. 4 show that DEfLS could perform better than DE and DEaLS outperformed all other algorithms by far.

The Rastrigin function (f_{ras}) is separable and symmetric. Therefore, it was solvable by MS-DSCG algorithm and the hybridization of DSCG with some GS made the search performance even better. Again DE could solve f_{ras} but at a very low speed (Table 5). Since DSCG could solve f_{ras} easily, the performance of DEfLS was pretty good in solving this function (Fig.5). But the performance of DEaLS was significantly better than that of DEfLS in statistical measure.

Being non-separable and asymmetric, the Griewank function (f_{grw}) is the most difficult problem among the three multimodal functions considered so far. Because of its epistatic and asymmetric characteristics, MS-DSCG could not solve the function at all. However, hybridization of DSCG with other GS was helpful in locating the global optimum of this benchmark. Once again this function was tractable for DE but at a lower speed. The performance of DEaLS in solving this problem was slightly better than DEfLS in terms of required fitness evaluation. However, both ACMA5 and ACMA10 exhibited better performance compared to DE variants and the ACMA5 algorithm performed the best.

Rosenbrock is an asymmetric multimodal function with strong epistacity. The strong epistatic nature of this function makes it very challenging to any optimization algorithm. Therefore, all the algorithms considered here failed to locate the global optimum of this function. The data presented in Table 7 show that improper hybridization of LS with GS may cause the MA to perform even worse than the original GS alone. The similar observation has also been reported in [13]. As a result DEfLS perform even worse than canonical DE. However, among the all search algorithms the best performance was exhibited by DEaLS, though it could not locate the global optimum either. The better performance of DEaLS over DE and DE over DEfLS, highlights that intelligent and adaptive exploitation is necessary to solve this problem and mere hybridization of LS with GS may unnecessarily waste function evaluation and degrade the overall performance of the algorithm.

The high multimodality and non-separability make Weierstrass function (f_{wrs}) a challenge for both global search and local search algorithms. The MS-DSCG got easily trapped in the large number of basins and could not find the optimum solution. Tactless hybridization of DSCG with some EA could not help in directing the search towards global optimum. Therefore, using a fixed LS scheme the DEfLS algorithm could not locate the global optimum of the function. Even other adaptive MAs such as ACMA5 and ACMA10 failed to solve the problem. However, DE itself showed bril-

Table 5: Performance comparison for f_{ras} ($N = 30$) Error Value (Fitness Evaluation)

Algorithm	Best	Worst	Average	SD	Success
MS-DSCG	0.0 (239)	0.0 (34266)	0.0 (9645)	0.0 (8302)	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%
DE	0.00E+00 (211212)	0.00E+00 (229919)	0.00E+00 (219935.1)	0.00E+00 (4728.08)	100%
DEfLS	0.00E+00 (242)	0.00E+00 (6109)	0.00E+00 (1712.1)	0.00E+00 (1464.22)	100%
DEaLS	**0.00E+00 (233)**	**0.00E+00 (4448)**	**0.00E+00 (1091.2)**	**0.00E+00 (1051.81)**	**100%**

Table 6: Performance comparison for f_{grw} ($N = 30$) Error Value (Fitness Evaluation)

Algorithm	Best	Worst	Average	SD	Success
MS-DSCG	5.792934	35.850984	12.249027	4.521426	0%
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%
DE	0.00E+00 (93688)	0.00E+00 (111072)	0.00E+00 (99695.0)	0.00E+00 (3829.43)	100%
DEfLS	0.00E+00 (3223)	0.00E+00 (53113)	0.00E+00 (23858.9)	0.00E+00 (11618.48)	100%
DEaLS	0.00E+00 (2600)	0.00E+00 (52493)	0.00E+00 (20296.3)	0.00E+00 (14432.84)	100%

liant performance by pinpointing the global optimum in every trial runs. But when DE was hybridized with static LS the DEfLS failed to locate the optimum in any trial. The possible reason behind the failure of DEfLS is the wasting of fitness evaluation by DSCG in fruitless exploitation. But in DEaLS the adaptive LS was successful in forming a synergy with DE and located the global optimum though at a higher cost of fitness evaluation compared to DE.

The convergence curves of Fig. 8 can explain these results better. Looking at the convergence curves of DE and DEaLS it was found that both of the curve have exactly the same shape but with a deflection along the x-axis. This is because starting with the same parameter settings DEfLS and DEaLS show similar search performance at the beginning of the search as the convergence curves suggest in Fig. 8. However, since DSCG was performing poor compared to DE, our adaptive scheme gradually reduced its CPU allocation close to zero. Then only the global search explored the search space and eventually showed DE like search characteristics. However, while the algorithm adjusts its parameters some fitness evaluations were wasted which cause DEaLS to converge using more fitness evaluation. Nevertheless, Fig. 8 helps to establish our claim that the proposed adaptive MA can favor GS or LS based on their online search performance.

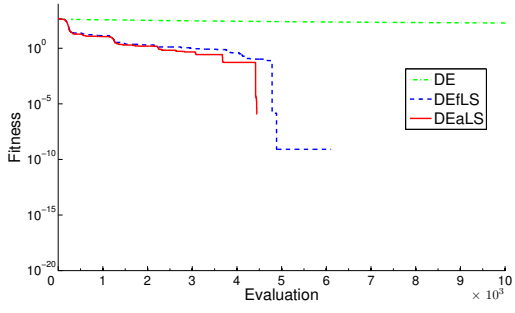

Figure 5: Convergence curve for f_{ras} function.

The Schaffer function (f_{scf}) is another non-separable problem with high multimodality. It was not solvable using MS-DSCG or the canonical DE algorithm. And the performance of DEfLS was poor compared to DE. The analogous relationship among the graphs in Fig. 8 and 9 suggests that once again DEaLS was successful to favor DE over LS and hence DEaLS performed better than DEfLS. Nevertheless, the best performance in terms of error value was exhibited by ACMA10 although none of the algorithms could find the global optimum.

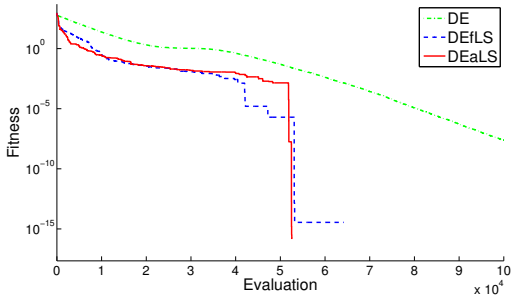

Figure 6: Convergence curve for f_{grw} function.

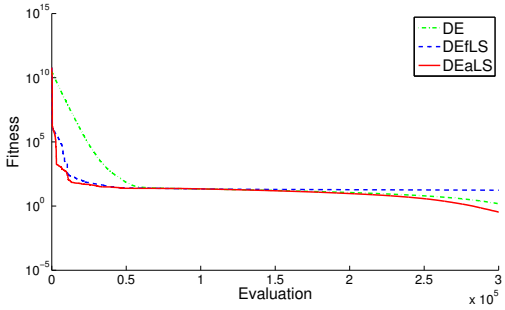

Figure 7: Convergence curve for f_{ros} function.

The f_{fms} problem is highly multimodal and strongly epistatic. Therefore, even this low dimensional problem put a challenge all optimization algorithms. DE could solve this problem in 16 runs out of 50 trials. But when DSCG was hybridized with DE in a static fashion, the performance of DEfLS dropped drastically in terms of robustness. The search traces in Fig.10 suggest that due to too much exploitation possibly the algorithm converged too fast. However, when

Table 7: Performance comparison for f_{ros} ($N = 30$) Error Value (Fitness Evaluation)

Algorithm	Best	Worst	Average	SD	Success
MS-DSCG	145.614786	2681.761437	681.762756	390.338469	0%
ACMA5	0.000072	16.184286	2.288657	3.677642	0%
ACMA10	0.000466	15.473135	1.230378	2.738209	0%
DE	2.70E-02	4.16E+00	1.52E+00	1.09E+00	0%
DEfLS	5.56E+00	2.09E+01	1.71E+01	3.22E+00	0%
DEaLS	**2.65E-08**	**3.99E+00**	**3.40E-01**	**7.53E-01**	**0%**

Table 8: Performance comparison for f_{wrs} ($N = 30$) Error Value (Fitness Evaluation)

Algorithm	Best	Worst	Average	SD	Success
MS-DSCG	35.269832	41.501266	39.144839	1.360618	0%
ACMA5	0.002177	0.897336	0.035927	0.161454	0%
ACMA10	0.001824	0.986958	0.097758	0.287558	0%
DE	**0.00E+00 (165204)**	**0.00E+00 (171159)**	**0.00E+00 (168748.9)**	**0.00E+00 (1449.98)**	**100%**
DEfLS	6.65E-02	1.06E-01	8.52E-02	7.69E-03	0%
DEaLS	0.00E+00 (223485)	0.00E+00 (232261)	0.00E+00 (227641.7)	0.00E+00 (2015.75)	100%

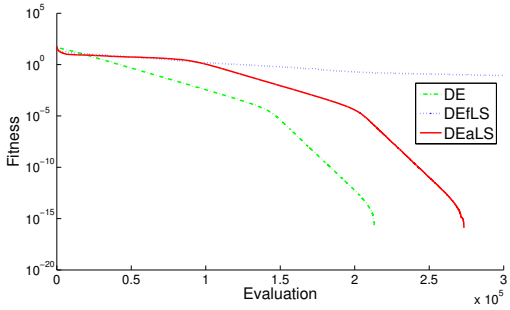

Figure 8: Convergence curve for f_{wrs} function.

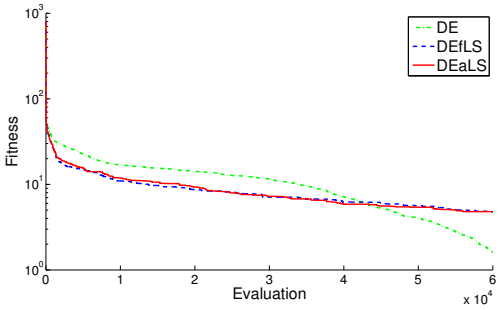

Figure 10: Convergence curve for f_{fms} function.

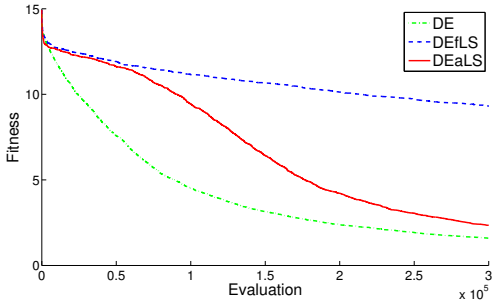

Figure 9: Convergence curve for f_{scf} function.

DSCG was incorporated with DE in a adaptive scheme the performance of DEaLS was comparatively better than DEfLS though still worse than the classic DE. The adaptive scheme was successful to mediate between the GS and LS and could find a better synergy between them compared to arbitrary hybridization. Nevertheless, both ACMA10 and ACMA5 showed better performance compared to DE variants in solving f_{fms}.

We also examined the results using statistical analysis (Student's t-test) at the 99% confidence level. In our analysis DEaLS was found significantly better than DEfLS in 8 benchmark functions and in other 2 benchmarks (f_{grw}, f_{fms}) no significant difference was observed between these two algorithms. Again comparing with DE, DEaLS was found significantly better in 7 benchmark functions. DEaLS was also significantly better than ACMA10 and ACMA5 in 7 and 6 benchmarks functions, respectively.

5. CONCLUSION

An adaptive memetic algorithms should be capable of adjusting its degree of exploration and exploitation. In this work we propose an local search (LS) scheme that tunes its extent of LS based on its online performance. In other words, based on its relative performance with GS, the number of individuals to undergo LS and the intensity of LS on them are adapted. This adaptation reciprocally favors GS or LS and thereby tries to find a synergy between them.

The LS scheme was implemented using DSCG and was hybridized with differential evolution. The resulted memetic algorithm, called DEaLS, was evaluated using ten standard benchmark problems. The results show that the DEaLS algorithm can favor its DE or DSCG components based on their online performance. Performance comparison with the canonical DE and some other MAs also highlight the superiority of the newly proposed algorithm.

In future, the adaptation policy, the adaptation rate and other parameters should be studied in greater detail to make the algorithm more robust to a wider range of problems.

6. REFERENCES

[1] N. K. Bambha, S. S. Bhattacharyya, J. Teich, and E. Zitzler. Systematic integration of parameterized

Table 9: Performance comparison for f_{scf} ($N = 30$) Error Value (Fitness Evaluation)

Algorithm	Best	Worst	Average	SD	Success
MS-DSCG	0.671650	1.840983	1.294955	0.274535	0%
ACMA5	0.038864	0.936715	0.430312	0.223992	0%
ACMA10	**0.097159**	**1.276680**	**0.371344**	**0.213785**	**0%**
DE	1.05E+00	1.85E+00	1.59E+00	1.50E-01	0%
DEfLS	7.73E+00	9.90E+00	9.30E+00	3.88E-01	0%
DEaLS	1.67E+00	3.53E+00	2.35E+00	3.37E-01	0%

Table 10: Performance comparison for f_{fms} ($N = 6$) Error Value (Fitness Evaluation)

Algorithm	Best	Worst	Average	SD	Success
MS-DSCG	0.049222	17.229564	6.981543	4.753076	0%
ACMA5	0.0 (4884)	21.483843	4.168509	6.189833	66%
ACMA10	**0.0 (6621)**	**17.382780**	**3.679700**	**5.595474**	**68%**
DE	4.76E-18 (44474)	11.57 (59609)	1.62208 (53256.8)	3.04530 (4726.60)	32%
DEfLS	1.55E-15 (23345)	15.29 (50013)	4.5833 (37392.8)	3.8631 (13789.96)	8%
DEaLS	6.68E-19 (5296)	14.37 (57411)	4.8025 (37520.2)	4.0943 (19379.00)	24%

local search into evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 8(2):137–155, 2004.

[2] A. Caponio, G. Cascella, F. Neri, N. Salvatore, and M. Sumner. A fast adaptive memetic algorithm for online and offline control design of pmsm drives. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 37(1):28–41, 2007.

[3] B. Freisleben and P. Merz. A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 616–621, 1996.

[4] W. E. Hart. *Adaptive Global Optimization with Local Search*. PhD thesis, University of California, California, 1994.

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

[6] N. Krasnogor and J. Smith. Emergence of profitable search strategies based on a simple inheritance mechanism. In *Proc. of the Genetic and Evolutionary Computation Conference*, pages 432–439, 2001.

[7] F. Lobo and D. Goldberg. Decision making in a hybrid genetic algorithm. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 121–125, 1997.

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

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

[10] G. M. Morris, D. S. Goodsell, R. S. Halliday, R. Huey, W. E. Hart, R. K. Belew, and A. J. Olson. Automated docking using a lamarckian genetic algorithm and an empirical binding free energy function. *Journal of Computational Chemistry*, 19(14):1639–1662, 1998.

[11] P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. In *Technical Report 826, Caltech Concurrent Computation Program. California Institute of Technology, Pasadena, California*, 1989.

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

[13] Q. H. Nguyen, Y.-S. Ong, M. H. Lim, and N. Krasnogor. Adaptive cellular memetic algorithms. *Evolutionary Computation*, 137(2):231–256, 2009.

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

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

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

[17] K. V. Price, R. M. Storn, and J. A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Springer, Berlin, Heidelberg, 2005.

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

[19] R. Storn and K. V. Price. Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, ICSI, 1995.

[20] R. Storn and K. V. Price. Differential evolution Ű a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

[21] D. Whitley, V. S. Gordon, and K. Mathias. Lamarckian evolution, the baldwin effect and function optimization. In *Proceedings of the Parallel Problem Solving From Nature (PPSN)*, pages 5–15, 1994.