A new GA-Local Search Hybrid for Continuous Optimization Based on Multi Level Single Linkage Clustering

Grégory Seront
Département d'Informatique
Université Libre de Bruxelles
50, av. Fr. Roosevelt
1050 Bruxelles BELGIUM

Hugues Bersini
IRIDIA
Université Libre de Bruxelles
50, av. Fr. Roosevelt
1050 Bruxelles BELGIUM

Abstract

Hybrid algorithms formed by the combination of Genetic Algorithms with Local Search methods provide increased performances when compared to real coded GA or Local Search alone. However, the cost of Local Search can be rather high. In this paper we present a new hybrid algorithm which reduces the total cost of local search by avoiding the start of the method in basins of attraction where a local optimum has already been discovered. Additionally, the clustering information can be used to help the maintenance of diversity within the population.

1 INTRODUCTION

The main strength of Genetic Algorithm is their global optimization capacities. They do not get easily trapped into local optima. On the Other hand, they are rather slow on fine local search. This weakness can be overcome by the combination (hybridization) of GA with a specific Local (LS) h method [6,4] like Powell's direction set or conjugate gradients.

In this context, each newly generated point *x* is replaced by the corresponding local optimum *LS(x)* found after the start of the search method from *x*.

This combination proved to increase greatly the performances compared to standard real coded GA. [4]. However, the cost of local search method can be high especially when the derivative of the function are not available. The usual cure for this problem is to reduce the cost of a single search by not allowing it to run until full convergence. This way, the local search produces an approximation of the local optimum which will be refined on later generation as it is depicted in Figure 1.

In this paper we present another approach based on the fact that it is unnecessary to run twice a Local Search on the same basin of attraction since the same local optimum will be rediscovered (in Figure 1, we see that the points *a*, *b* and *c* lead to the same local optimum). In our method, points belonging to a basin are detected by a clustering algorithm, and only one LS is started in each basin. This has the advantage of reducing the total cost of Local Search while providing precise local optima. In the Section 2, we explain the clustering method we take as basis for our hybrid, then in section 3 we explain a first idea on how to hybridize GA and clustering. In Section 4, we describe a modification we did to the clustering method in order to overcome a diversity maintenance problem that may appear in a naive hybridization. In Section 5 we address the problems that may arise due to the non-uniformity of the sampling. The Section 6 present some simulation results and the Section 7 concludes.

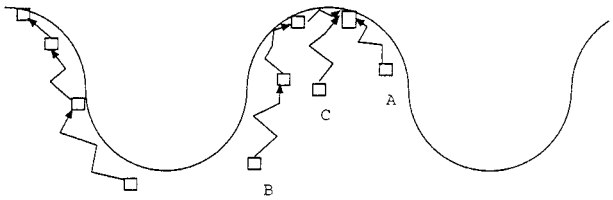


Figure 1: Multiple LS Steps from Generations to Generations

2 CLUSTERING METHOD

As we said, our goal is to reduce the total cost of Local Search by noting that there is no need to perform a Local Search in a basin where another LS has already been performed since it would lead to an already discovered Local Optimum.

A whole family of global optimization methods knows as clustering methods do exactly this [7]. They sample the search space, identify points belonging potentially to the same basin of attraction and use this information to perform only one LS in each basin of attraction.

In this paper, we use the Multi Level Single Linkage (MLSL) clustering method, because it proved to be both theoretically and empirically the best one [5].

The algorithm for MLSL is shown below. It constructs a set of local optima X^*. At each iteration k, we generate N new points $x_{(k-1)N+1}, \ldots, x_{kN}$ with a uniform distribution. Then we search for points that do not have better neighbor with a threshold distance r_k (we give the value of r_k below). These points are chosen as starting points for LS.

$k = 1$; $X^* = \emptyset$
repeat
  Generate N uniformly distributed and points $x_{(k-1)N+1}, \ldots, x_{kN}$
  Rank the x_i by increasing $f(x_i)$
  for $\forall i | 1 \leq i \leq kN$ do
    if $\nexists j | f(x_j) < f(x_i)$ and $\|x_j - x_i\| < r_k$ then
      $X^* \cup = LS(x_i)$
    end if
  end for
until $k > MaxK$

These points are good candidates for LS as we see in Figure 2, which depicts a clustering situation. The rounded points are those upon which LS is performed: an arrow from *a* to *b* means that LS will not be performed upon *a* because *f(a)>f(b)* (if we minimize) and the distance between *a* and *b* is less than r_k. We see that if r_k is large enough, at most one LS will be performed in each basin; and if r_k is small enough (smaller than the distance between two local optima) LS will be performed within every basin with a sample point.

The critical distance is computed in such a way that the probability of not having a point within distance r_k is equal to an *ak* that will be controlled during the run.

If a region is sampled uniformly, we can approximate the probability that one point of the sample has no other point of the sample within a distance *d* by

$$\alpha_k = \left(1 - \frac{d^n \pi^{n/2}}{\Gamma(1 + n/2) m(S)}\right)^{kN-1}$$

Where *Γ(n)* is the continuous version of the factorial of *n-1*This approximation comes from the fact that for a small *d*, the probability that a single point falls within a distance *d* is the volume of the sphere of radius *d* divided by *m(s)* (*m(S)* is the measure of the search space) Noting that we have *kN-1*other points, the probability of not having a point within distance *d* is this probability exponent *kN-1*.

The critical distance r_k must be so that the probability of not having a point within this distance is less than α_k if the region is sampled uniformly. If the closest point is farther than r_k we can reject the hypothesis that the region is sampled uniformly with a confidence *1-*α_k

Extracting *d* from 1, we have

$$r_k = \left(\frac{\Gamma(1 + \frac{1}{2}n) m(S)}{\pi^{n/2}} (1 - \alpha_k^{1/(kN-1)})\right)^{1/n}.$$

$$\alpha_k = \left(1 - \sigma \frac{\log kN}{kN}\right)^{kN-1},$$

$$r_k = \left(\frac{\Gamma(1 + \frac{1}{2}n) m(S)}{\pi^{n/2}} \frac{\sigma \log(kN)}{kN}\right)^{1/n},$$

so that as $k \to \infty$, r_k and $\alpha_k \to 0$ and thus the clustering becomes more and more precise as *k* increases.

3 CLUSTERING GA: FIRST IDEA

With GA and MLSL, we have all the needed elements to construct an efficient global optimization algorithm: the GA is able to sample efficiently the search space and the MLSL is able to detect efficiently local optima. We must now decide how to combine them.

The combination of GA with the MLSL clustering algorithm is straightforward at the level of MLSL. We keep the MLSL algorithm as it is with the only exception that instead of using a uniform distribution to generation the sample, we use a GA.

At the level of GA, many options are available. The first naive idea is to use the clustering method just as a way to improve LS by avoiding unneeded computation. The GA provides a sample of points $x_1, \ldots x_n$ to MLSL which returns the local optima corresponding to each of these points, $LS(x_1), \ldots LS(x_n)$ and each *xi* is replace by its corresponding local optimum $LS(x_l)$. Here is the macro-algorithm for this method.

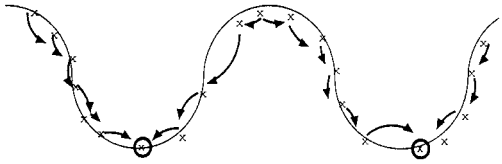


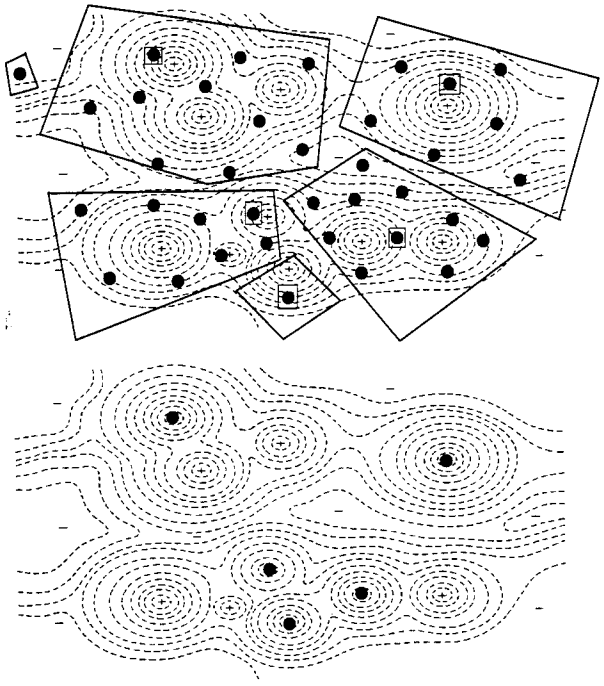
Figure 2: Clustering in MLSL. LS will only be applied on rounded point.

$k = 1; X^* = \emptyset$
repeat
  Generate N uniformly distributed and points $x_{(k-1)N+1}, \ldots, x_{kN}$
  Rank the x_i by increasing $f(x_i)$
  for $\forall i | 1 \leq i \leq kN$ do
    if $\nexists j | f(x_j) < f(x_i)$ and $\|x_j - x_i\| < r_k$ then
      $X^* \cup = LS(x_i)$
    end if
  end for
until $k > MaxK$

In this case, the GA replaces the sampling procedure, but the population of GA remains independent from the past samples used by the clustering algorithm. We thus have two distinct populations. One of fixed size for the GA and another one for MLSL, which contains all, the past points generated.

This naive approach is not applicable as it is. Indeed, during the first generation, the density of the sample of the search space is not very high and a cluster can encompass several basins of attraction. In this case, only one local optimum will be detected in each group of basins and all the points inside a same cluster will be replaced by the same local optima creating a huge loss of diversity as it is depicted in Figure 3 where the plain lines represent the clusters boundaries.

Obviously, something as to be done about the maintenance of diversity. The next section addresses this problem.



4 MAINTENANCE OF DIVERSITY

Diversity must be maintained at two different levels: locally, at the level of the cluster, we must avoid the concentration of all the points of a cluster toward a single point. Globally, we must avoid the convergence of the GA toward a single cluster. To solve these problems, we do not replace the points by their corresponding local optima anymore. Instead, we apply a two level selection strategy using the information provided by the clustering process.

At the local level, we apply an intra-cluster selection step on each cluster independently. Each cluster is treated as a separate population. Selection is applied inside each cluster as in a normal GA process using the fitness of the points without local optimization. The only exception is that the worst point of a cluster is replaced by the local optima of the cluster if it is not yet present. After this step, the number of points belonging to each cluster inside the GA population has not changed.

At a global level, we will change this number. Selection now acts globally, taking as fitness for a point *x* belonging to a cluster *C* the fitness of the local optima detected inside *C*. This process can be seen as treating the cluster as selection unit and changing their relative weight inside the GA population. To avoid global premature convergence, a fraction of the population is filled we already discovered local optima.

With these modifications, we have the following algorithm where $C(x_j)$ is the cluster to x_j is assigned and $L(C)$ is the local optimum found inside the cluster C and P the population of the GA.

$P \leftarrow N$ Uniformly sampled points
repeat
  Add the points of P to MLSL samples
  Associate each individual of P to a cluster
  !Intra Cluster Selection!
  for all Clusters C_i present in P do
    Let I the worst individual of P belonging to C_i
    Replace I by the local optima of C_i
    Perform local selection on the points of C_i belonging to P using $f(x)$ as fitness
  end for
  $OldPop \leftarrow P$; $P \leftarrow \emptyset$
  !Cluster diversification!
  !w ≡ number of local minima detected by MLSL!
  for i=1 to $min(w, \nu N)$ do
    Select a cluster C_i
    $P \cup \leftarrow (LS(C_i), f(LS(C_i)))$
  end for
  !Global Selection!
  Select $N - min(w, \nu N)$ points x_j from $OldPop$ using $f(LS(C(x_j)))$ as fitness
  Apply genetic operators to produce the new population P
until Stopping Criterion

With these modifications, we already have a good algorithm, however in some case a further refinement could be necessary as we explain in the next section.

5 THE NON UNIFORMITY OF THE SAMPLE

During our presentation of MLSL, we saw that it relied on a uniform sampling of the search space. Our hybridization of GA with MLSL results indeed in a MLSL with the sampling distribution generation by the GA and thus different from a uniform one. One may wonder what is the effect of this non-uniformity on the clustering process and on the remarkable theoretical properties of MLSL. Intuitively, the non-uniformity of the sample does not prevent the global minimum to be discovered provided that the probability of generation a point in its basin is different from zero.

What the MLSL does to decide if LS will be applied on a point is detect that there is a descending path, formed by sample points distant of at most r_k, connecting a sample point to another one for which LS has already been performed (see Figure 2). If r_k is small enough, all the points in the path belongs to the same basin. As r_k tends toward *0* with increasing *k*, if *k* is large enough, this will be the case.

The non-uniformity of the sample does not change anything to this. As long as at least a point is generated in the basin of local optimum, it will be discovered once *k* gets large enough. Hence, we can replace the uniform distribution by another one biased toward regions of the search space where we hope the global optima is more likely to be.

On the other hand, the non-uniformity has implication on the probability with which LS is applied to a given sample point. With uniform distribution, r_k is computed in such a way that the probability of not having a point within distance r_k is *ak=(logkN)/N*. With non-uniform distribution, the r_k is too large in high density of sampling regions and too small in low-density regions. Since regions sampled with a low density can be seen as „less promising“ in GA view, we have the paradoxical effect of performing LS with a higher probability on points sampled in less promising regions. Sample points are thus wasted in high-density regions and LS are wasted in low-density regions.

5.1 RE-CLUSTERING

A solution to this would be to estimate the density of points locally in a region and deduce a local r_k.

To do this, we designed a method that we called re-clustering. It is a two-phase process:

1) Clustering is applied to the whole sample with a global r_k
2) Local Clustering is performed on each cluster formed after phase 1 with a critical distance r_{Ci} local to each cluster *Ci* , and with sample formed by the points of this cluster.

Each r_{Ci} is a function of the density *Di* of points within the cluster *Ci*. To estimate *Di*, we count the number of points included in a sphere of radius r_k centered at the local optima of the cluster.

We then start from the expression giving the critical distance r_t as a function of the number of points *t* and of the size of the search space *m(S)*:

$$r_t = \left(\frac{\Gamma(1 + \frac{1}{2}n) m(S)}{\pi^{n/2}} \frac{\sigma \log(t)}{t} \right)^{1/n},$$

noting that *t/m(S)* is the density of the sampling, we obtain the critical distance r_{Ci}, local to the cluster C_i:

$$r_{C_i} = \pi^{-1/2} \left(\Gamma(1 + n/2).\sigma \frac{\log(m(S).D_i)}{D_i} \right)^{1/n}$$

With this modification, the hybridized algorithm remains the same at the exception that the clustering now includes a re-clustering phase.

6 EXPERIMENTAL RESULTS

The algorithm we described in the preceding sections is only a framework. Many choices remain to be done concerning the operators used inside the GA. For our implementation, we use here the real coded GA of Michalewicz as a basis (see [3]).

We compared our new hybrid (GA-MLSL) with its most logical competitors: a Multi Level Single Linkage algorithm (MLSL), a GA with LS performed by Powell’s method run until convergence (GA-LS), and a GA with LS performed by a single step of the Powell’s method (GA-LowLS). All the algorithms tested in this paper rely for their local search upon Powell’s direction set, which is a derivative free method.

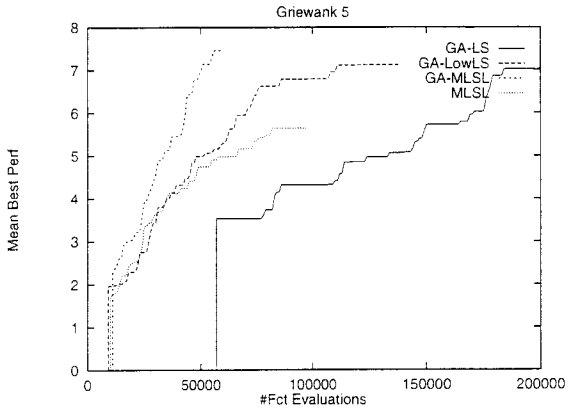


Figure 4: Mean best perf for Griewank with $N = 5$ and $D = 4000$

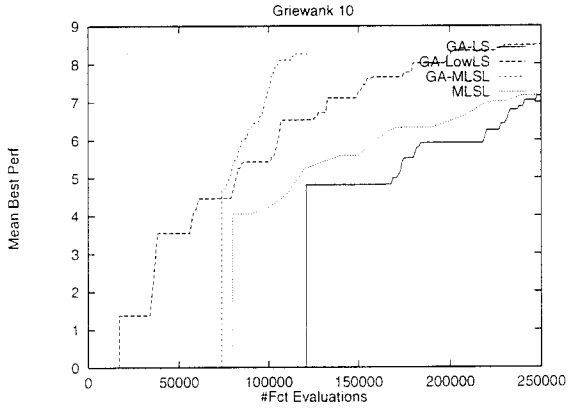


Figure 5: Mean best perf for Griewank with $N = 10$ and $D = 4000$

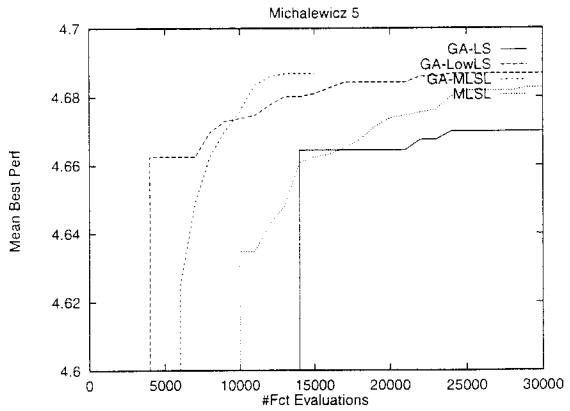


Figure 6: Mean best perf for Michalewicz with $N = 5$ and $m = 10$

We tested these algorithms on the function set defined for the first International Contest on Evolutionary Optimization (ICEO) [1]. These are the five and ten dimensional versions of the Griewank function (with *D=4000*), the Michalewicz function (with *m=10*) and the Langerman's function.

These function posses a panel of characteristics interesting in the context of global optimization: The Griewank function posses a global quadratic structure upon which cosinusoidal noise is added. The Michalewicz function is a separable, which possesses *N!* local optima if the separability property is not exploited and the Langerman function is formed by 5 waves which interact with each other forming a multitude of local optima.

We show in figure 4 trough 9 the evaluation of the best individual averaged over 50 runs. We see that on every function at the exception of the five dimensional version of the Langerman, the GA-MLSL is the best performing algorithm in the long run. However, if all that is required is a low precision, GA-LowLS is often able to produce faster good solutions. This is the logical consequence of two different LS cost reduction policy. In one case the precision is lower in the other the density of sample of the search space exploited is lower.

7 CONCLUSION

In this paper we showed that the cost of Local Search in a GA-LS hybrid could be reduced with a clustering method by avoiding multiple rediscoveries of the local optima. In addition, this clustering method supplies informations that can be used to maintain the diversity in the population.

Many other ways to hybridize GA with clustering method exists. The algorithm we present here is only a study of feasibility.

References

[1] H. Bersini, M. Dorigo, G. Seront, S. Langerman and L. Gambardella. Result of the first International Contest on Evolutionary Optimization (ICEO) In *[8] 1996*

[2] Richard Brent *Algorithms for minimization without derivatives.* Prentice-Hall 1973.

[3] Z. Michalewicz. *Genetic Algorithms + Data Structure = Evolution Programs.* Springer Verlag, 1992.

[4] H. Mulhenbein, M. Schomisch and J. Born. The Parallel Genetic Algortihm as Function Optimizer. In *Proceedings of the fourth international conference on genetic algorithm* 1991

[5] A.H.G. Rinnoy and G.T. Timmer. Stochastic Global Optimization Methods part ii: Multi-level methods. *Mathematical Programming,* 1987

[6] G. Seront, H. Bersini. Simplex GA and Hybrid Methods. In *1996 International Conference on Evolutionary Computation* 1996.

[7] Aimo Torn and Antanas Zilinskas. *Global Optimization* Springer-Verlarg 1989

[8] Michalewiz Z., editor, *Proceeding of the first IEEE conference on Evolutionary Computation* IEEE, 1994

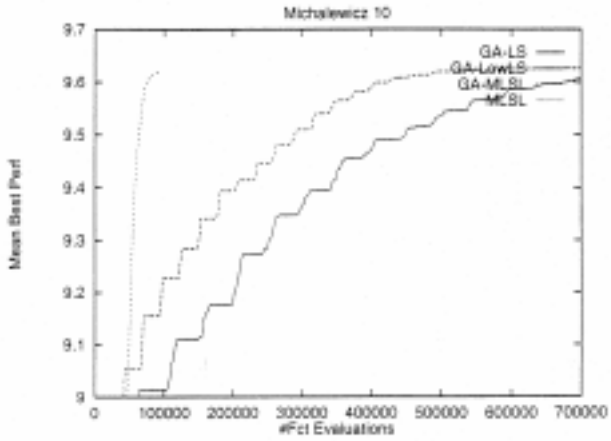


Figure 7: Mean best perf for Michalewicz with $N = 10$ and $m = 10$

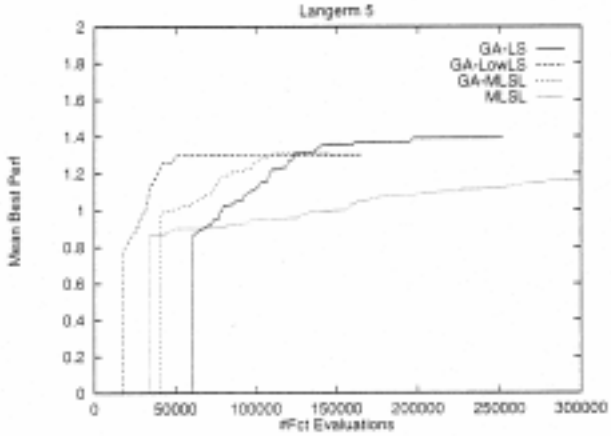


Figure 8: Mean best perf for Langerman with $N = 5$ and $m = 5$

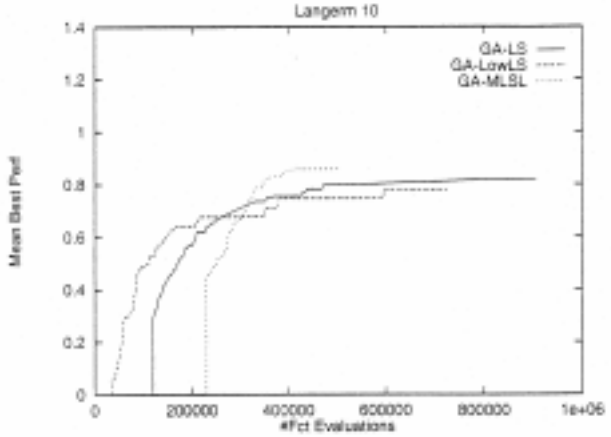


Figure 9: Mean best perf for Langerman with $N = 10$ and $m = 5$