# Automatic Graph Drawing and Stochastic Hill Climbing

**Alejandro Rosete-Suárez**
CEIS, ISPJAE, Marianao 19390
C. Habana, CUBA
rosete@ceis.ispjae.edu.cu
Tel/Fax: (537) 27.15.75

**Alberto Ochoa-Rodríguez**
Center of Artificial Intelligence.
La Habana, 10400, CUBA
ochoa@cidet.icmf.inf.cu
Tel/Fax: (537) 33.33.73

**Michele Sebag**
LMS, Ecole Polytechnique,
91128 Palaiseau Cedex, FRANCE
sebag@cmapx.polytechnique.fr
Tel/Fax: (33) 01.69.33.30.26

## Abstract

In the literature of Evolutionary Computation, it is very strange to find papers where the results of Evolutionary Algorithms are compared to other algorithms. Stochastic Hill Climbing is a simple optimization algorithm that has shown a competitive performance with respect to many powerful algorithms in the solution of different problems. It has also outperformed some Evolutionary Algorithms in previous papers. Here we fairly review some of these papers. We also compare many Evolutionary Algorithms in the context of Graph Drawing. Graph Drawing addresses the problem of finding a representation of a graph that satisfies a given aesthetic objective. This problem has many practical applications in many fields such as Software Engineering, and VLSI Design. Our results demonstrate that Stochastic Hill Climbing is also the best algorithm in this context. We give some general guidelines in order to explain our results. Our explanations are based on landscape characteristics.

## 1 INTRODUCTION

In the literature of Evolutionary Computation, it is very strange to find papers where the results of Evolutionary Algorithms (EA) are compared to other algorithms. Stochastic Hill Climbing (SHC) is a simple optimization algorithm that has shown a competitive performance with respect to many powerful algorithms in the solution of different problems. It has also outperformed some Evolutionary Algorithms in previous papers. Here we fairly review some of them. This review (in Section 2) shows that the niche of problems where SHC is a good algorithm seems to be wider than the common belief.

This kind of results is somewhat unexpected because almost everybody thinks that this simple algorithm is not very useful. In general, it is hard to identify "a priori" which algorithm will be the best for a specific task. We hope that this paper will induce the authors of EA papers to use SHC as a baseline method in order to show EA power [Juels and Wattenberg, 1994].

In this paper, we also compare many Evolutionary Algorithms in the context of Graph Drawing (Section 3). Graph Drawing addresses the problem of finding a representation of a graph that satisfies a given aesthetic objective. The representation of a graph is often given by an embedding of its nodes in a target grid. This problem has many practical applications in many fields such as Software Engineering, and VLSI Design. An extended survey on this topic is [Battista et al., 1994]. Some papers have explored the suitability of EA in Graph Drawing problems. Some good examples can be found in [Kosak et al. 1991; Branke et al., 1997; Rosete-Suárez and Ochoa-Rodríguez, 1998, Tettamanzi 1998]. Our results demonstrate that Stochastic Hill Climbing is also the best algorithm in this context.

In Section 4, we give some general guidelines in order to explain our results. Explanations are based on landscape characteristics. Also, Fitness-Distance Correlation factor is computed. Section 5 contains our principal conclusions.

## 2 STOCHASTIC HILL CLIMBING AS A BASELINE METHOD

Stochastic Hill Climbing (SHC) is one of the simplest optimization algorithms. SHC keeps a population of only one individual. The initial individual is often selected in a random way. The most popular SHC strategy consists of producing a random variation in the current individual in a similar way as mutation works. Then, the new individual is compared to the old one. If the new one is better (or at least equal) than the old solution, then the new solution replaces the old one. This process is repeated until a maximum number of evaluations are done.

Some versions of this simple algorithm have been developed. For instance, Multiple SHC is a version of SHC that restarts the search after a given number of fitness evaluations are completed. An extended discussion of versions of SHC is in [Baluja, 1995; Jones, 1995; Juels and Wattenberg, 1994; Yuret, 1994].

Two basic version of SHC are:

- Next Ascent SHC (NA-SHC): It works as it was explained in the previous paragraph. NA-SHC explores the neighborhood until a solution equal or better than the current one is found. Then, the algorithm takes this new solution as the current one.
- Best Ascent SHC (BA-SHC): It is very similar to Artificial Intelligence's HC. Best ascent SHC explores all the neighborhood of the current solution, chooses the best neighbor solution, and takes it as the current one.

Now, we will review some papers showing how a simple SHC algorithm outperforms some EA in different contexts. Our aim is to show that the set of problems where this phenomenon occurs seems to be wider than the common belief. We do not claim that this list of papers is exhaustive. In spite of this fact, we think that our review is sufficient for supporting our analysis. If there are other papers that show this phenomenon, they will support our objective. Here, we will discuss the results of some important papers that give theoretical and practical evidence of the power of this simple method.

We are not trying to state that SHC is better than EA. According to No Free Lunch Theorem for Search [Wolpert and Macready, 1995], all algorithms perform exactly equal when they are averaged over all possible functions. Therefore, it is incorrect to state that one algorithm is the best (or the worst) one. Consequently, SHC has also its niche of functions where it behaves efficiently. In many papers, the authors explain the selection of an EA to solve a problem by stating that their problem is very complex. Frequently, the complexity of the problem is not well explained. Besides, the comparisons to easy versions of SHC are rather unusual. This situation may bias new research to use Evolutionary Algorithms as the only (or at least the best) possibility. In spite of this extended creed, many papers have shown that some of the most popular functions optimized by means of an EA are easily solved by SHC. Without the intention to be exhaustive, we will mention some cases.

In [Davis, 1991], Davis shows that in De Jong's test functions and in two of Schaffer's functions, a simple SHC method outperforms two versions of Genetic Algorithms (GA): generational and steady-state. These functions cover a wide variety of interesting characteristics that an optimization method must face. The SHC method used only flips one bit in each step and accepts the new solution if its evaluation is equal or better than the previous one. The author also shows that a shift of the mapping function that translates the binary chromosome to the real decoded parameter affects the performance of all the algorithms. This modification provokes that GA outperforms SHC. The author claims that such a kind of analysis may conduct to new test functions. In a recent paper [Rana, 1998], it is shown that such a kind of shift of the mapping function alters the number of local optima. Besides, from the expressions derived in the same paper it is easy to see that if the number of neighbors increases, then the probability of being stuck in a local optimum may be reduced.

A relevant paper is [Juels and Wattenberg, 1994]. The authors compare SHC to GA. It is shown that SHC outperforms some GA taken from different papers on problems such as: maximum cut problem, Koza's 11-multiplexer problem, Multiprocessor Document Allocation Problem, and the job-shop problem. In the first problem, a single-bit-flipping SHC is used. In the other problems, more sophisticated neighborhood operators are used. The authors indicate that the size of the neighborhood affects the search. Some evidences are given for using a multi-restart SHC in order to improve the results. The main idea introduced in this paper is that SHC should be used as a baseline method in order to demonstrate the convenience of GA. Unfortunately, such a kind of test is not often used.

In [Ishibuchi et al., 1994], the authors show that Multi-Descent SHC outperforms GA and Taboo Search (TS) in many instances of fuzzy flow-shop scheduling problems. In addition, the performance of SHC is very similar to Simulated Annealing (SA) which is the best in many cases. The paper also claims for the convenience of hybridizing GA by using some local search technique. Similar observations are expressed in many other papers in order to obtain a better GA performance [Michalewicz et al., 1997].

In [Yuret, 1994], the performance of SHC is analyzed. In this thesis, three heuristics extend SHC algorithm.

1. The search is restarted (when a local optimum is reached) at a point very distant from the local optima that have been discovered before.
2. The size of the step of neighborhood operator is adjusted according to the recent progress of the search.
3. The directions of the recent success are probed with preference.

This algorithm is specially designed for exploring continuous spaces. The resulting SHC is compared to GA and Simulated Annealing (SA). It outperforms GA in five De Jong's functions and SA in four of them. In Traveling Salesman Problem, the performance of SHC is similar to SA (GA is not considered in this case). In a Refraction Tomography problem, GA outperforms SHC, but the hybrid method (combining GA and SHC) outperforms both. In other problems, SHC is more efficient than some specialized algorithms.

In [Baluja, 1995], seven algorithms are compared in different instances of some popular problems: job-shop scheduling, traveling salesman, knapsack, bin-packing, neural network weight optimization, and standard numerical optimization. In all cases, the algorithms do not incorporate any problem specific knowledge in order to improve the results. Three versions of SHC are used. They differ in two aspects. The first aspect is whether to accept or to reject the moves to the regions with the same evaluations. The second aspect is the number of evaluations that each algorithm made before the search is restarted. The performance of SHC versions is similar. In 16 of 27 cases, it is found beneficial to accept moves to regions with equal values. In all function, Gray codes improved the results. A theoretical explanation of this result is in [Rana, 1998]. Two versions of elitist GA are compared (the main difference is in term of parameters and operators used). In seven of 27 functions, one of the SHC is better than both GA versions. These seven functions are four instances of Traveling Salesman Problem (of four), one instance of bin-packing problem (of four), one instance of Neural Network weight optimization (of four), and one numerical function optimization (of six). In 14 of 27 functions, one instance of SHC is better than one of the instances of GA. In addition, SHC outperforms PBIL in some cases. PBIL is the best performing algorithm in general sense. SHC always uses one-bit flipping as neighborhood operator.

In [Whitley et al., 1995], one-bit flipping SHC is compared to some versions of GA in some numerical optimization functions. In this case, SHC is better than both an Elitist GA and Genitor GA when no interaction exists between the variables. This interaction depreciates the performance of SHC. This result is to be expected because only one bit is changed for exploring the neighborhood of the current solution and the variables are coded in many bits. It is also reported that a SHC that changes on average 2 bits does not perform better than the simple SHC. Because of the difficulty created by the codification and the interaction, this result is not very rare. The best algorithm is CHC; a GA that uses a very disruptive crossover that tries to recombine a selected individual with a very different one. Following the discussion in [Jones, 1995], this recombination operator may be acting as a form of macro-mutation. That seems to be convenient for problems where many interactions exist between the variables. In the paper [Whitley et al., 1995], the authors state that more complex local-search methods would perform better. In addition, the convenience of some hybrid methods is discussed.

In a recent paper [Sebag and Schoenauer, 1997], a study on socializing the action of many HC is conducted. In this paper the similarities between using many HC and using Evolution Strategies (ES) are discussed (despite the main interest in ES is devoted to continuous spaces and its performance in integer spaces is not very well guaranteed). In this paper, two types of SHC are developed: Natural Societies and Historical Societies. Both work as ES when many individuals performing SHC compete to become a member of the next generation. In addition, Historical Society uses a binary vector to guide the search. This is very similar to Baluja's PBIL [Baluja, 1995]. In the paper, the experiments conducted in [Baluja, 1995] are extended by introducing these new algorithms in the comparison. The new algorithms outperform (or at least are equal) in five of six 900-bits functions. The authors stated that these methods were run with different number of bits allowed to flip. In many cases, they outperform ES. However, they cleverly say that a great perturbation may distort the search.

As can be seen, SHC and other more complex algorithm based on local search have outperformed other Evolutionary Algorithms in many problems. In many cases, the SHC used is a very trivial algorithm where no attempt is done in order to improve it. According to these results, it is to be expected that the parameters that control the type of neighborhood used may affect drastically SHC performance. For instance, [Kingdom and Dekker, 1996] present a SHC algorithm that varies the codification used during the search process. The resulting algorithm is tested with very good results.

# 3 STOCHASTIC HILL CLIMBING IN GRAPH DRAWING

Graph Drawing is a very important task in many fields of research and development. Graph Drawing addresses the problem of finding a representation of a graph that satisfies a given aesthetic objective. The representation of a graph is often given by an embedding of its nodes in a target grid. This problem has many practical applications in many fields such as Software Engineering, VLSI Design and plant layout. An extended survey on this topic appears in [Battista et al., 1994].

Some papers have explored the suitability of EA in Graph Drawing problems. Some good examples can be found in [Kosak et al. 1991; Branke et al., 1997; Rosete-Suárez and Ochoa-Rodríguez, 1998, Tettamanzi 1998]. Examples of the aesthetic criteria that the objective function takes into account are minimizing line crossing, minimizing the global length of the lines representing edges, etc. Many good heuristic algorithms have been developed, but they are often specific to a certain kind of graph, a given aesthetic criteria, etc.

In a previous paper [Rosete-Suárez and Ochoa-Rodríguez, 1998] we show the convenience of using a general-purpose search method like Genetic Algorithms in order to solve this problem in a robust and general way. This robust and general approach it is necessary due to the growing number of applications that uses diagrams. In our previous study, GA permits to search for good solutions of different aesthetic criteria with a similar efficiency. A survey of theoretical and practical aspects of Graph Drawing is in [Battista et al., 1994].

Here, we present a comparison of some general-purpose algorithms. We compared the performance of Evolution Strategies, Genetic Algorithms [Michalewicz et al., 1997], Univariate Marginal Distribution Algorithms [Muhlenbein, 1998], and Stochastic Hill Climbing in the task of minimizing line crossing. No general algorithm has been developed for solving this problem [Battista et al., 1994]. Now, we will explain the experiments.

### 3.1 CODIFICATION

In our experiments, two codifications were used.

- Conventional Codification (CC): Each coordinate of the nodes is directly coded in an integer gene. Therefore, each individual consists of a string of integers like $X_1$, $X_2$, ... $X_i$ ... $X_n$, $Y_1$, $Y_2$, ... $Y_i$ ... $Y_n$ where each $X_i$ and $Y_i$ represents the position of the i-th node in X and Y-axis. The number of nodes of the graph is n. For each gene we allow n alleles. Thus, each n-node graph will be embedded in a n x n grid.
- Order Codification (OC): Each individual represents two permutations. These permutations establish the ordering of the nodes in each axis. Both permutations can be translated to CC by assigning a position to each node according to its position in the permutation. This translation is done for each axis. In this codification, each individual consists of a string like $OX_2$, ... $OX_i$ ...$OX_n$, $OY_2$, ... $OY_i$ ...$OY_n$. Here, n is also the number of nodes. Each gene $OX_i$ represents the relative position in X-axis of the i-th node amongst the first i nodes. Therefore, $OX_i$ has i possible values. That is the reason why the first node is not coded (i.e., values for $OX_1$ and $OY_1$ are not given). If i=1, only one value is possible for this node. The translation process is done node-by-node from the second node to the n-th. Each step of the algorithm consists of the insertion of one node in the partial ordering of its previous nodes. This process will systematically produce partial orderings in each axis. This process is done for each coordinate independently.

Now, we give an example in order to illustrate OC. See Figure 1. If n=3, the individual "1321" (that is, $OX_2$=1, $OX_3$=3, $OY_2$=2, $OY_3$=1) can be translated into the individual "213231" using CC (that is, $X_1$=2, $X_2$=1, $X_3$=3, $Y_1$=2, $Y_2$=3, $Y_3$=1). The partial orderings are represented in Figure 1 in gray boxes. In X-axis, the first ordering is "21" because $OX_2$=1 (i.e. node 2 will be placed in the first position between the first two (i=2) nodes). After, the third node is inserted in the third position (as $OX_3$=3). This produces the final ordering in X-axis "213". This ordering is then translated to the absolute positions $X_1$=2 (the second in the ordering), $X_2$=1 (first) and $X_3$=3 (third). The same process is done in Y-axis obtaining the final ordering "312". This algorithm is taken from [Ordoñez-Reinoso and Valenzuela-Rendón, 1992].

We use this code in order to alleviate the "competing conventions" problem. This problem was observed in [Branke et al., 1997]. It consists of the many ways where it is possible to code similar drawing. For instance, if the positions of all nodes of a graph are shifted the same amount in any direction, then the resulting drawing will have the same quality (in aesthetic criteria evaluation) despite the difference in the coordinates values.

The "competing conventions" problem in CC arrives when a recombination operator is used. For instances, two identical solutions, only differing in shifts or rotations, are very different in terms of chromosomes. Figure 2 show how a shift and reflection of a drawing provoke a very different chromosome. Consequently, it may cause an overlapped diagram in the offspring after crossover. This overlapped diagram is often worse than its parents are.

We hope that OC alleviates this problem because the drawings that only differ due to shifting will be represented in the same way. This is caused by the fact that all these cases are represented by the same permutation.

In [Branke et al., 1997], this problem is partially solved by using a modified crossover operator that aligns two drawings before recombining them. As we will use a recombinative algorithm as UMDA where many parent are used for producing new children, it is difficult to use such a kind of modified recombination operator. That is the reason why we use OC.

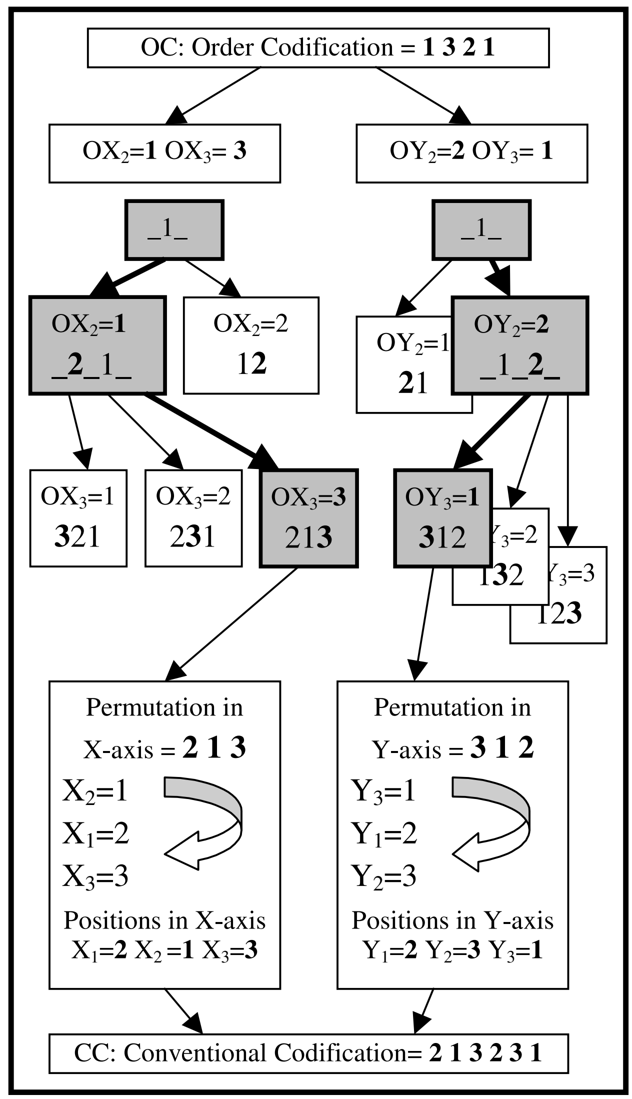


Figure 1: An example of a translation from OC to CC

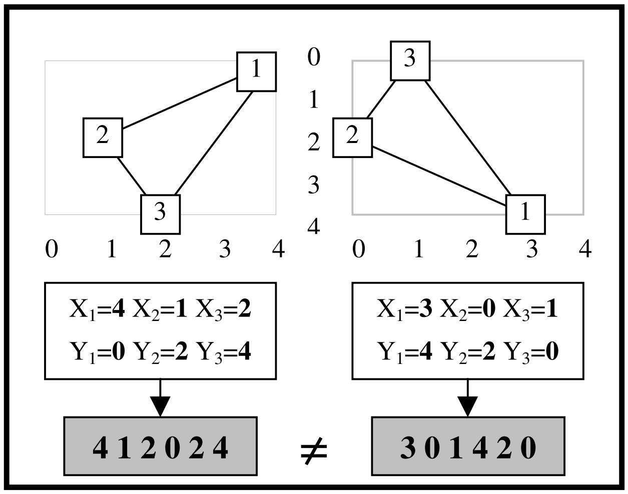


Figure 2: An example of competing conventions problem.

Despite its advantages, OC does not solve the "competing conventions" problems produced by reflection and rotation.

### 3.2 ALGORITHMS

In the experiments, we compared the following Evolutionary Algorithms:

- Evolution Strategies (ES) [Michalewicz et al., 1997]. Here, we used ES without recombination, as is the common practice in ES community. As we use integer genes, we can not use Gaussian real-value mutation. Mutation is performed by randomly selecting one of the possible allele values. In each mutation, only one gene is changed. Preliminary experiments augmenting the number of genes allowed to be mutated provoke worse results. We use $(\mu+\lambda)$-ES, with four different parameter settings:
  - $\mu=1$, $\lambda=1$ what it is exactly equal to SHC with next ascent strategy (NA-SHC)
  - $\mu=1$, $\lambda=5$ what it is very similar to SHC with best ascent strategy (BA-SHC)
  - $\mu=7$, $\lambda=50$
  - $\mu=10$, $\lambda=100$
- Univariate Marginal Distribution Algorithms (UMDA) [Muhlenbein, 1998]. This algorithm is one of the first members of the Marginal Distribution Algorithms. UMDA generates the next population by setting each allele in a probabilistic way. Each allele will be selected according to its frequency in the pool of individuals obtained in the selection process from the N individuals in the population. Selection is performed by Truncation Selection (only the T best individuals will be deterministically chosen). Advantages of UMDA over GA are discussed in [Muhlenbein, 1998]. Two types of UMDA were compared:
  - UMDA as was previously described, with T parameter equal to 20 and N equal to 100.
  - UMDA modified in such a way that the selected individuals are picked from the current generation and the parent pool. This modification is very similar to $(\mu+\lambda)$-ES compared to $(\mu,\lambda)$-ES [Michalewicz et al., 1997]. T parameter is 100, and N is 100. This setting selects all the individuals in the first generation, but after the selection pressures becomes effective.

The third and fourth parameter settings in $(\mu+\lambda)$-ES are very commonly used. For UMDA, the settings used were obtained through a comparison of the results with the combination of N=50, 100, 200 and 400; and T= 10, 20, 50 and 100 for both versions.

We also conducted experiments with a simple GA with elitism and many parameter settings, but in all cases it performs worse than UMDA. We do not show these results here.

The Evolutionary Algorithms described represent an acceptable range of the Evolutionary paradigm, including recombinative algorithms (UMDA and GA) and mutation-based algorithms. Two population-free algorithms are included (1,1)-ES and (1,5)-ES. The second perform a greater exploration of the neighborhood of the current solution before changing it. As (1+1)-ES is equal to NA-SHC, and (1+5)-ES is similar to BA-SHC, we are using ES as a framework for evaluating SHC performance.

### 3.3 BENCHMARK

The size of the possible instances of the problem is infinite due to the infinite number of aesthetic criteria that can be defined [Battista et al., 1994,Rosete-Suárez and Ochoa-Rodríguez, 1998], and the infinite number of graphs

Therefore, we must restrict our comparison. As our main goal is to develop a graphical tool for solving graph drawing tasks, we compared the algorithms in the context of the most widely used aesthetic criterion: minimizing line-crossing. In spite of only using this aesthetic criterion (minimization of line-crossings), the problem remains hard (NP-Hard [Battista et al., 1994]).

We define three sets of graphs: graphs with 10 nodes, graphs with 20 nodes, and graphs with 30 nodes. These numbers of nodes are very common in graphical interfaces context. We generate 20 random graphs in each set.

In Graph Drawing problem for interactive applications, it is very important to achieve results in very short time intervals, then we stop each algorithm after a few number of fitness evaluations. We compared the performance of each one of these algorithms after 1000, 2000, and 3000 evaluations.

### 3.4 RESULTS

In Table 1, 2 and 3 we show the result of the comparison of the six algorithms (with CC and OC) in terms of their results after 1000, 2000 and 3000 fitness evaluation. We did 20 independent runs for each algorithm trying to solve each graph.

Table 1: Ranking of the algorithms after 1000 evaluations

| ALGORITHMS | CC | OC |
|---|---|---|
| (1+1) - ES (SHC) | 1 | 3 |
| (1+5) – ES | 2 | 4 |
| (7+50) – ES | 5 | 6 |
| (10+100) – ES | 7 | 8 |
| UMDA (100, 20) | 12 | 11 |
| UMDA (100, 100) | 10 | 9 |

Table 2: Ranking of the algorithms after 2000 evaluations

| ALGORITHMS | CC | OC |
|---|---|---|
| (1+1) - ES (SHC) | 1 | 5 |
| (1+5) – ES | 2 | 6 |
| (7+50) – ES | 3 | 4 |
| (10+100) – ES | 7 | 8 |
| UMDA (100, 20) | 12 | 11 |
| UMDA (100, 100) | 10 | 9 |

Table 3: Ranking of the algorithms after 3000 evaluations

| ALGORITHMS | CC | OC |
|---|---|---|
| (1+1) - ES (SHC) | 1 | 6 |
| (1+5) – ES | 2 | 7 |
| (7+50) – ES | 3 | 5 |
| (10+100) – ES | 4 | 9 |
| UMDA (100, 20) | 12 | 11 |
| UMDA (100, 100) | 10 | 8 |

We compare the algorithms in terms of the average of the numbers of line-crossings in the best drawing obtained (in each run). The ranking of each algorithm is computed in each graph. Then, algorithms are ranked according their rankings in all the set of graphs. The resulting ranking in the set of 20-node graphs is shown in Table 1, 2 and 3.

Similar results were obtained for the set of 10-node and 30-node graphs. In the set of 10-node graphs all algorithms perform very well. Besides, these rankings are almost regularly obtained in each specific graph.

According to the results obtained, we can draw some conclusions:

- The algorithms compared herein may be grouped according to their performance in three sets:
  - Winner set: (1+1)-ES, (1+5)-ES, both with CC are remarkably the winners.
  - Intermediate set: (7,50)-ES (both codes); (1+1)-ES and (1+5)-ES, both with OC; and (10+100)-ES with CC.
  - Loser set: (10+100)-ES with OC, UMDA and GA (with both codes)
- A fair glance at the rate of line-crossing reduction achieved by the "winners" and "losers" with respect to the maximum possible value is shown in Table 4. It shows the average of this rate. For the "intermediate" set of algorithms, the values are in the range between the "winners" and "losers". As graphs were generated at random, we do not know which is the optimum in each case. For example, non-planar graphs can not be drawn without crossing. Then, rates of 0% are often impossible.

Table 4: Reduction of the number of line-crossings by the winners and the losers.

| | 20-nodes graphs | | | 30-nodes graphs | | |
|---|---|---|---|---|---|---|
| Thousands of evaluations | 1 | 2 | 3 | 1 | 2 | 3 |
| Winners | 5 | 4 | 4 | 8 | 7 | 7 |
| Losers | 13 | 10 | 10 | 16 | 13 | 10 |

- SHC permits to obtain the better results using CC. As OC is principally developed for overcoming a problem of crossover, it is expected that OC does not improve the results of SHC. The same conclusion is valid for the other ES variants. Results validate this statement.
- OC improves the results obtained by using recombinative algorithms (the two UMDA versions).

- In this problem, ES does not improve its results by augmenting the size of μ and λ. On the contrary, results become worse when μ and λ increase.
- Other remarkable result (partially shown in Table 4) is the comparative performance of SHC with respect to other algorithms with more evaluations. Results obtained by SHC with 1000 evaluations and using CC are better than all other algorithms and codes with 2000 evaluations. In addition, results of SHC with 1000 evaluations are better than many of the other algorithms with 3000 evaluations. ES with CC is the only exception.

# 4 DISCUSSION

As it is correctly stated in [Wolpert and Macready, 1995], there is no best algorithm for solving all types of problems. Furthermore, if one algorithm outperforms other algorithms in a given problem, it may be expected the opposite results in other problems.

In spite of the reasons given in [Juels and Wattenberg, 1994], and of the practical results described in Section 2, many authors make no effort in order to know whether their problems may be solved using a simple method like SHC. As it was explained in Section 3, in our experiments we obtain these unexpected results. Therefore, here we will outline some reasons in order to explain them.

As it is explained in [Jones, 1995], the concept of landscape depends on the fitness function and the operators (or codification) used. Line-crossing landscape seems to be highly multimodal. Despite the fact that a more theoretical study must be done for accepting or rejecting this statement, we believe that this expected difficulty is alleviated in this case (SHC, with CC in line-crossing minimization) due to two reasons.

First, in the space there are many points with equal evaluations that are neighbors, because not all the changes in the coordinate produce a new crossing or eliminate it. This kind of region is usually called a "plateau" [Jones, 1995]. Such regions allow SHC to have access to better points to continue its improvements. Such regions are caused by the redundancy introduced by CC for this problem. Examples of these regions are those of the drawings only differing in one coordinate if the difference in the node positions does not alter the evaluation. We think that it may be the cause of the poor performance introduced by OC in SHC. OC reduces such a kind of region because many redundancies of CC are eliminated.

The second reason is the existence of many local optima that are equally good. The principal problem of multimodality is that of trapping SHC in local optima. We think that many global optima exist in line-crossing landscape. For example, whether an optimal drawing is shifted, rotated or reflected it will be optimal. Besides, it is expected that the same kind of "plateau" as explained before must exist in the neighborhood of each optimum. In addition, because of the great size of the neighborhood defined by the mutation operator in this context, it will be expected that the number of local optima be drastically reduced. This relationship is theoretically studied in [Rana, 1998]. In this case, mutation landscape in CC is defined by n*(n-1) points, because we can mutate one of the n genes to one of the remaining n-1 alleles.

In addition, it would be expected that the remaining local optima would have very good values. Thus, these local optima will serve also for our purposes. In real-world graph-drawing applications, a good (not optimal) solution is also useful if it is obtained in a short time.

In order to give a further evidence of the characteristics of the fitness landscape we calculated the Fitness Distance Correlation (FDC). FDC was originally developed to serve as a predictive measure of GA performance. However, it must be better for mutation-based algorithms because FDC considers the distance in mutation landscape. FDC consists of the correlation factor between the distance from a solution to the next optimum and its fitness value. If both are very correlated then the search will be easy. See [Jones, 1995] for a further discussion on this subject.

Here, in order to compute FDC we performed a random sampling in the search space defined by CC. Then, for each solution we computed the number of different genes between this solution and the best of the solutions obtained in the sampling. This value is used as a distance measure. We performed many random independent random sampling and we averaged these results of FDC. The average value of this FDC is -0.34 what says that the landscape is easy for the mutation operator used.

In addition, this result may be still better. In this case, the existence of reflections, shifts, and rotations of the best solution may produce some equally good solution with different chromosomes. It must be expected that one of these chromosomes be closer to the other solutions than the best solution considered. As we did not consider this possibility, the value of FDC is affected. Also, "plateau" regions affect the correlation factor as they represent regions where distance varies but fitness remains constant. However, they do not affect the search because of the reasons previously explained. We believe that FDC confirms the suitable characteristics of this fitness-operator landscape for SHC.

Here, we are not claiming for using SHC for obtaining the global optimum for this kind of problem. The small number of fitness evaluations may be a further cause of why SHC outperforms other algorithms in this problem. SHC exploits its current state sacrificing a wider search. Here, our experiments sustain the convenience of such a local search for obtaining good solutions very quickly. As this is our benchmark, we try neither to extend nor to reject the convenience of SHC for different objectives.

In future works, we will study the convenience of using non-random initialization methods. In addition, we think that it must be necessary to make experiments with other kinds of mutation operators. Besides, it must be studied whether it is convenient to restart the search from different initial positions.

## 5 CONCLUSIONS

In this paper, we compared many evolutionary algorithms in a very popular graph drawing problem: line-crossing minimization. Experiments conducted with random graphs of different size show that a simple Stochastic Hill Climbing algorithm outperforms very efficient and popular population-based optimization algorithms such as Evolution Strategies and Genetic Algorithms. We discussed our results based on the characteristics of the landscape defined by the combination of codification, operator, and fitness function. We also outline some directions for future research. In addition, an introductory section describes early works where Stochastic Hill Climbing has outperformed some Evolutionary Algorithms. We hope that our experience will serve to understand the necessity of comparing Evolutionary Algorithms to simple algorithms in order to demonstrate the convenience of a method in a specific problem. Such a comparison is very useful in real world applications.

**Acknowledgements**

This research was partially developed when the first author was at LRI, University of Paris-Sud (XI), France. The authors would like to thank the suggestions and comments of the anonymous reviewers.

**References**

Baluja, S.: An Empirical Comparison of Seven Iterative and Evolutionary Function Optimization Heuristics, Technical Report CMU-CS-95-193, Carnegie-Mellon University, 1995.

Battista, G.D., Eades, P., Tamassia, R., Tollis, I.G.: Algorithms for drawing graphs: an annotated bibliography, Discrete Geometry: Theory and Applications, 4:235-282,1994.

Branke, J., Bucher, F., Schmeck, H.: A genetic Algorithm for drawing undirected graphs, Proceedings of Third Nordic Workshop on Genetic Algorithms and their Applications, Alander, J.T. (editor), pp.193-206, Vaasa, 1997 (available at ftp://ftp.aifb.uni-karlsruhe.de/pub/br/gagd.ps.gz)

Davis, L.: Bit-Climbing, Representational Bias, and Test Suite Design, Proceedings of the 4th International Conference on Genetic Algorithms, pp.18-23, Morgan Kaufmann, San Mateo, 1991.

Ishibuchi, H., Yamamoto, N., Murata, T., Tanaka, H.: Genetic Algorithms and neighborhood search algorithms for fuzzy flowshop scheduling problems, Fuzzy Set and Systems, 67, pp.81-100, 1994.

Jones, T.C.: Evolutionary Algorithms, Fitness Landscapes and Search, Ph. D. Thesis, University of New Mexico, Alburquerque, 1995.

Juels, A., Wattenberg, M.: Stochastic Hill-climbing as a Baseline Method for Evaluating Genetic Algorithms, Technical Report, University of California at Berkeley, 1994.

Kingdom, J., Dekker, L.: Morphic Search Strategies, Proceedings of First IEEE International Conference on Evolutionary Computation, ICEC'96, Nagoya, 1996.

Kosak, C., Marks, J., Shieber, S.: A Parallel Genetic Algorithm for Network-Diagram Layout, Proceeding of the 4th International Conference on Genetic Algorithms, pp.458-465, 1991.

Michalewicz, Z., Hinterding, R., Michalewicz, M.: Evolutionary Algorithms, Fuzzy Evolutionary Computation (Chapter 2), Pedryczs, W. (ed.), Kluwer Academic, 1997.

Muhlenbein, H.: The equation for Response to Selection and its Use for Prediction, Evolutionary Computation, 5, pp. 303-346, 1998.

Ordoñez-Reinoso, Y., Valenzuela-Rendón, M.: Optimización de Permutaciones con Algoritmos Genéticos - El Problema del Vendedor Viajero, Proceedings of 3rd Congreso Iberoamericano de Inteligencia Artificial, Habana, pp. 271-282, Noriega Editores, 1992.

Rana, S., Whitley, L.D.: Search, Binary Representation and Counting Optima, Proceedings of Workshop on Evolutionary Algorithms, Sponsored by the Institute for Mathematics and its Application, (in press), GENITOR Group, Colorado State University, 1998.

Rosete-Suárez,A., Ochoa-Rodríguez,A.: Genetic Graph Drawing, Proceeding of the 13th International Conference of Applications of Artificial Intelligence in Engineering, Galway, pp.37-41, 1998.

Sebag, M., Schoenauer, M.: A Society of Hill Climbers, Procedings of the 4th IEEE International Conference on Evolutionary Computation, ICEC'97, Indiana, 1997.

Tettamanzi, A.G.B.: Drawing graphs with Evolutionary Algorithms, Proceedings of 1998 Conference on Adaptive Computing in Design and Manufacture, ACDM'98, Plymouth, April 1998 (to appear).

Whitley, D., Beveridge, R., Graves, C., Mathias, K.: Test Driving Three 1995 Genetic Algorithms: New Test Functions and Geometric Matching, Journal of Heuristics, 1: 77-104, 1995.

Wolpert, D.H., Macready, W.G.: No Free Lunch Theorem for Search, Technical Report SFI-TR-95-02-010, Santa Fe Institute, 1995.

Yuret, D.: From Genetic Algorithms To Efficient Optimization, M. Sc. Thesis, Department Electrical Engineering and Computer Science, MIT, 1994.