Stochastic Reverse Hillclimbing and Iterated Local Search

Carlos Cotta
Dept. Lenguajes y Ciencias de la Computación, University of Málaga
Campus de Teatinos (3.2.49)
29071-Málaga (Spain)
ccottap@lcc.uma.es

Enrique Alba
Dept. Lenguajes y Ciencias de la Computación, University of Málaga
Campus de Teatinos (3.2.12)
29071-Málaga (Spain)
eat@lcc.uma.es

José M[a] Troya
Dept. Lenguajes y Ciencias de la Computación, University of Málaga
Campus de Teatinos (3.2.14)
29071-Málaga (Spain)
troya@lcc.uma.es

Abstract- This paper analyzes the detection of stagnation states in iterated local search algorithms. This is done considering elements such as the population size, the length of the encoding and the number of observed non-improving iterations. This analysis isolates the features of the target problem within one parameter for which three different estimations are given: two static a priori estimations and a dynamic approach. In the latter case, a stochastic reverse hillclimbing algorithm is used to extract information from the fitness landscape. The applicability of these estimations is studied and exemplified on different problems.

1 Introduction

The reverse hillclimbing (RHC) algorithm (designed by Jones and Rawlins [JR93]) is a very adequate tool for studying fitness landscapes. This algorithm can be used for determining the basin of attraction of a desired point of the search space. However, one of drawbacks of this algorithm is the fact that, in some situations, the size of this basin of attraction may be very large (even of the same magnitude than the whole search space). This is especifically true in smooth landscapes. In fact, and as pointed out by the authors, the more rugged the fitness landscape, the more efficient the algorithm is.

This work presents a modification of the reverse hillclimbing algorithm with application to iterated local search. To be precise, the algorithm is used to extract some measures of the fitness landscape. These measures are subsequently utilized to determine when to terminate the execution of the algorithm. This is done by calculating the probability of stagnation provided that the algorithm has not yielded better solutions for a certain number of evaluations. Since local search algorithms are not appropriate for very rugged landscapes, the use of this stochastic version of the RHC algorithm is justified.

The remainder of the article is organized as follows: first, a mathematical analysis is done in section 2 to determine the probability of stagnation of a local search algorithm as a function of the parameters of the algorithm. This analysis isolates the features of the objective problem into one measure for which different estimations are given in section 3. First, two static estimations are discussed in subsection 3.1. Then, a dynamic estimation is presented in subsection 3.2, based on the use of a stochastic reverse hillclimbing algorithm. These estimations are evaluated with respect to some common termination criteria in section 4. Finally, some conclusions are outlined in section 5.

2 A Probabilistic Analysis of Stagnation

The following analysis is valid for discrete $(\mu + \lambda)$-LS algorithms. These algorithms maintain a population of μ individuals, create new λ individuals in each iteration and select the best μ individuals from both the current and the newly created populations for the next generation. Due to its elitist behavior, it can be easily seen that the population will remain unchanged if the algorithm has stagnated. Hence, the analysis is based on this observation, i.e., we will calculate the probability of stagnation conditioned to the population remaining unchanged for a certain number of iterations.

Initially, suppose that $\mu = 1$ and $\lambda = 1$. Assume that no new individual has been accepted in the population after r iterations. The probability of this situation (denoted by E_r) is

$$P(E_r) = \sum_{i=0}^{\mathcal{M}_{max}} P(\mathcal{M}_i) \cdot P(E_r/\mathcal{M}_i) = \quad (1)$$

$$= \sum_{i=0}^{\mathcal{M}_{max}} P(\mathcal{M}_i) \cdot \left(\frac{\mathcal{M}_{max} - i}{\mathcal{M}_{max}} \right)^r \quad (2)$$

where $\mathcal{M}_{max}$ is the total number of ways in which an individual can be mutated and $P(\mathcal{M}_i)$ is the probability that i mutations produce an individual better than, at least, another individual in the population. The last term represents the conditioned probability of none of these mutations being performed after r iterations. The measure $P(\mathcal{M}_i)$ is very important since it carries all the information about the problem being solved. Section 3 is entirely devoted to discuss several estimations of this parameter.

Now, since $P(E_r/\mathcal{M}_0) = 1$, using Bayes' Theorem to

calculate the stagnation probability $P(\mathcal{M}_0/E_r)$ yields

$$P(\mathcal{M}_0/E_r) = \frac{P(\mathcal{M}_0)}{P(E_r)} \quad (3)$$

In a more general situation, $\mu > 1$. In this situation, $P(\mathcal{M}_i^\mu)$ is used to denote the probability of a total number of i mutations being capable of improving individuals in the population. This value can be used to modify equation (2) yielding

$$P(E_r) = \sum_{i=0}^{\mu\cdot\mathcal{M}_{max}} P(\mathcal{M}_i^\mu)\cdot P(E_r/\mathcal{M}_i^\mu) = \quad (4)$$

$$= \sum_{i=0}^{\mu\cdot\mathcal{M}_{max}} P(\mathcal{M}_i^\mu)\cdot P(E_1/\mathcal{M}_i^\mu)^r \quad (5)$$

where $P(E_1/\mathcal{M}_i^\mu)$ is the probability of none of the i possible mutations being applied in a single step. This value can be defined as

$$P(E_1/\mathcal{M}_i^\mu) = \sum_{j=1}^{\mu} P_{sel}(j)\cdot\sum_{k=0}^{m(i)} P(E_1/\mathcal{M}_i^\mu, j, k) \quad (6)$$

$$P(E_1/\mathcal{M}_i^\mu, j, k) = P(\mathcal{M}_{k,j}^\mu/\mathcal{M}_i^\mu)\cdot\left(\frac{\mathcal{M}_{max}-k}{\mathcal{M}_{max}}\right) \quad (7)$$

In the above expressions, $P_{sel}(j)$ is the probability of selecting the jth individual in the population, $m(i) = \min(i, \mathcal{M}_{max})$, and $P(\mathcal{M}_{k,j}^\mu/\mathcal{M}_i^\mu)$ represents the probability of k mutations producing an acceptable individual when applied to this jth individual given a total number of i acceptable mutations. The former depends upon the selection mechanism used (fitness proportionate [Gol89], tournament [BT95], etc.). As to the latter, it can be calculated as

$$P(\mathcal{M}_{k,j}^\mu/\mathcal{M}_i^\mu) = \frac{P(\mathcal{M}_{k,j}^\mu \cap \mathcal{M}_i^\mu)}{P(\mathcal{M}_i^\mu)} \quad (8)$$

Then, $P(\mathcal{M}_0^\mu/E_r)$ can be calculated as shown in equation (3) using the value obtained with equation (5). Finally, notice that no improvement being achieved after r iterations with $\lambda > 1$ is equivalent to consider $\lambda' = 1$ and $r' = r\cdot\lambda$. Thus, for given values of μ and $\mathcal{M}_{max}$, the value of r can be calculated so as to ensure with desired confidence that the algorithm has stagnated.

3 Three Estimations of $P(M_i)$

As stated in the previous section, $P(M_i)$ is a central parameter in the analysis. Three different estimations are given in this section. The first and the second ones are static, i.e., they do not consider the past history of the algorithm. The third one is more sophisticated and includes knowledge about both the evolution of the algorithm and the features of the induced fitness landscape.

3.1 Static Estimations

The first and easiest estimation that can be given for $P(\mathcal{M}_i)$ is to consider a uniform distribution across all problems and populations. This constitutes a totally general estimation in the line of the assumptions of the "No Free Lunch" Theorem [WM95]. Hence, it can be used when no knowledge about the problem being solved is available. In this situation, $P(\mathcal{M}_i^\mu)$ is given by

$$P(\mathcal{M}_i^\mu) = C_i^{\mu\cdot\mathcal{M}_{max}}\cdot\left[\sum_{j=0}^{\mu\cdot\mathcal{M}_{max}} C_j^{\mu\cdot\mathcal{M}_{max}}\right]^{-1} = \quad (9)$$

$$= C_i^{\mu\cdot\mathcal{M}_{max}}\cdot 2^{-\mu\cdot\mathcal{M}_{max}} \quad (10)$$

where C_i^j is the number of ways in which a subset of i elements can be extracted from a set of j elements. With the above estimation, equation (8) can be rewritten as

$$P(\mathcal{M}_{k,j}^\mu/\mathcal{M}_i^\mu) = \frac{C_k^{\mathcal{M}_{max}}\cdot C_{i-k}^{(\mu-1)\cdot\mathcal{M}_{max}}}{C_i^{\mu\cdot\mathcal{M}_{max}}} \quad (11)$$

An example of the application of this estimation is given below.

Example 1.
Consider the Schwefel function [WGM94]. This is a nonlinear multidimensional bounded minimization problem whose functional form is as follows:

$$F(\vec{x}) = nV + \sum_{i=1}^{n} -x_i\sin(\sqrt{|x_i|}), x \in [-512, 511] \quad (12)$$

where $V = -\min\{-x\sin(\sqrt{|x|}) : x \in [-512, 511]\}$. The graphic aspect of this function can be seen in Fig. 1 (top-left).

Now, consider a (1+10)-LS algorithm using an integer representation and a mutation operator that increases or decreases by a fixed ammount a randomly selected vector position (wrapping is performed if necessary). Then, if the target problem is an n-dimensional Schwefel function, the number of mutations per individual is $\mathcal{M}_{max} = 2n$. With these values for μ and $\mathcal{M}_{max}$, equations (10) and (11) can be used to calculate $P(\mathcal{M}_0^\mu/E_r)$ according to equation (3). The resulting stagnation probabilities are shown in Fig. 1 for different dimensionalities. An experimental example is also shown.

This assumption is quite simple and may be the only available option in many circumstances. However, it can be improved. Consider that $P(E_r)$ is defined as a weighted sum of different hypothesis (number of acceptable mutations) about the state of the algorithm. The weight of each hypothesis is the value of $P(\mathcal{M}_i^\mu)$.

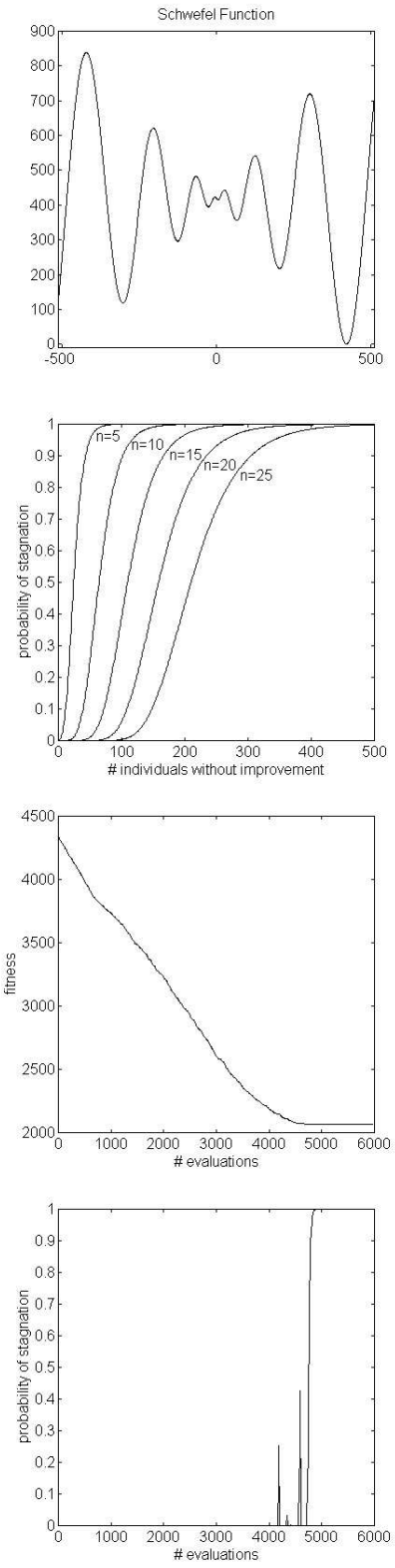

Figure 1: From top to bottom, graphic aspect of the basic Schwefel function, stagnation probabilities for different dimensionalities, evolution of fitness in a run of the algorithm, and evolution of the stagnation probability in the same run.

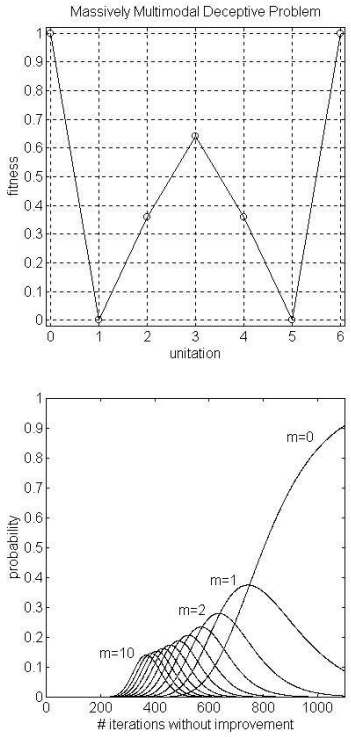

Figure 2: Graphic aspect of the Massively Multimodal Deceptive Problem (left) and evolution of different hypothesis (0 to 10 acceptable mutations) as a function of the number of non-improving iterations in a 25-MMDP (right).

If enough problem knowledge is available, these weights can be better estimated. To be precise, $P(\mathcal{M}_i)$ could be calculated as $P(\mathcal{S}_i)$, where $\mathcal{S}_i$ represents the subset of the search space for which i acceptable mutations exist. Then, $P(\mathcal{M}_i^\mu)$ would be

$$P(\mathcal{M}_i^\mu) = \sum_{j_1+\ldots+j_\mu=i} \prod_{k=1}^{\mu} P(\mathcal{M}_{j_k}) \qquad (13)$$

It must be noted that, while determining such subsets is generally difficult, their relative weight (and hence their probability) can be easily calculated either analytically or empirically. An analytical example is shown below (empirical examples are deferred to subsection 3.2).

Example 2.

Consider the *Massively Multimodal Deceptive Problem* [Gol95]. This problem is defined by the concatenation of k 6-bit segments. The fitness of a string is determined by the sum of each segment fitness, which is calculated as shown in Fig. 2 (top). Using equations (10) and (11) yields the scenario depicted in Fig. 2 (bottom): as the number of non-improving iterations increases, different hypothesis are discarded and, finally, only stagnation remains plausible as a hypothesis.

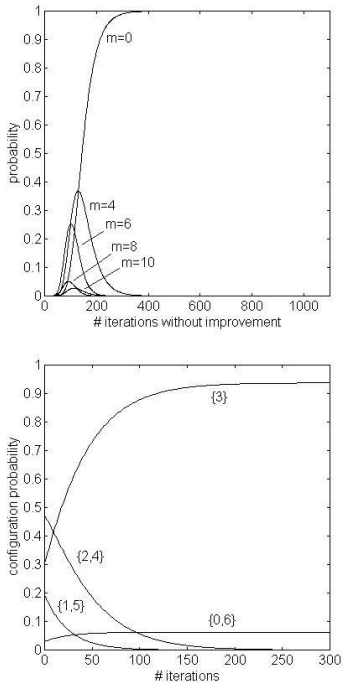

Figure 3: Evolution of some feasible hypothesis (0 to 10 acceptable mutations) as a function of the number of non-improving iterations in a 25-MMDP (left) and theoretical evolution of unitation probabilities in a segment of the same problem (right).

Notice that all hypothesis (i.e., number of improving mutations ranging from 0 to $\mu \cdot \mathcal{M}_{max}$) are taken into account in the first estimation. However, this is not always realistic. As an example, consider a (1+10)-LS algorithm using single bit-flip mutation. It can be seen that a new individual will be accepted only if it has a better objective value. This requires at least one segment to have 1,2,4 or 5 ones, i.e., a non-optimal configuration that can be improved.

There exist six acceptable mutations for segments with one or five 1s, and four mutations for segments with two or four 1s. Let α (resp. β) be the number of segments with one or five (resp. two or four) ones in a given string. Then the number of acceptable mutations is $6\alpha + 4\beta$, $(\alpha + \beta \leq k)$. Since there are 12 possible α-configurations ($C_1^6 + C_5^6$) and 30 possible β-configurations ($C_2^6 + C_4^6$), the probability of that situation is

$$P(S^{\alpha,\beta}) = \frac{C_\alpha^k \cdot C_\beta^{k-\alpha} \cdot 12^\alpha \cdot 30^\beta \cdot 22^{k-\alpha-\beta}}{2^{6k}} \qquad (14)$$

Then, $P(\mathcal{M}_i)$ can be calculated as $\sum_{6\alpha+4\beta=i} P(S^{\alpha,\beta})$. Notice that $P(\mathcal{M}_i)$ will be greater than zero only if i can be decomposed as $6\alpha + 4\beta$. Subsequently, some hypothesis will not be tested (e.g., $\mathcal{M}_1$) since they are not feasible. This is shown in Fig. 3 (top).

Some comments must be done about this second estimation. Firstly, notice that, in spite of an analytical study as above be not always possible, empirical tools exist that allow extracting the necessary information as shown in next subsection. Secondly, if $\mu > 1$ some non strictly-improving mutations could produce an acceptable individual. Since hypothesis stating a high number of acceptable mutations are quickly discarded, this is not a critical issue in practice. Nevertheless, stagnation probabilities will be henceforth referred to single individuals to keep the analysis consistent with this fact. For the sake of simplicity, the population is considered stagnated when all individuals have stagnated (i.e., they are at a local optimum). This will be complemented with a dynamic study of the evolution of each individual.

This study is related to a third consideration. Equation (14) statically estimates the probability of each configuration. However, these probabilities evolve with time. This is illustrated in Fig. 3 (bottom) which shows the theoretical evolution of these probabilities using a Markov-chain analysis. Clearly, a more accurate analysis must take this evolution into account. This is done in next subsection.

3.2 Dynamic Estimation

As stated above, the dynamic evolution of the algorithm must be considered to obtain a more accurate measure of the stagnation probability. This implies that $P(\mathcal{M}_i/\mathcal{I}_n)$ (where $\mathcal{I}_n$ represents that a certain individual has been improved n times) will be used instead of $P(\mathcal{M}_i)$. Following the reasoning mentioned in the previous section leads to

$$P(\mathcal{M}_i/\mathcal{I}_n) = \sum_{\alpha \in \mathcal{S}_i} P(\alpha/\mathcal{I}_n) \qquad (15)$$

Now, $P(\alpha/\mathcal{I}_n)$ can be calculated as

$$P(\alpha/\mathcal{I}_n) = \sum_{\beta \in \mathcal{B}(\alpha)} P(\beta) \cdot P(\beta \xrightarrow{n} \alpha) \qquad (16)$$

where $\mathcal{B}(\alpha)$ is the basin of attraction of α, and $P(\beta \xrightarrow{n} \alpha)$ is the probability of arriving at α after n improving steps.

These probabilities can be empirically found by means of Reverse Hillclimbing [JR93, Jon95]. However, notice that an exhaustive exploration is out of question since the basin of attraction of a given point may be larger than the number of atoms in the universe. Hence, a stochastic version of the algorithm is used. This stochastic algorithm (SRHC) does not perform a comprehensive exploration of the basin of attraction, but it carries out a parameterized sampling of this basis, subsequently extrapolating the results. This algorithm is used to obtain an approximation of the distribution of improving mutations as a function of the distance to the

target point (a bidimensional matrix `mf`), as well as the size of the basin of attraction of this point at different distances (a vector `tm`). The algorithm is based on the `Explore` function whose pseudocode is shown below.

```
function [mf', tm'] = Explore (p, op, mf, tm,
                               d, mv, w)

# p:   a point of the search space
# op:  a search operator
# mf:  bidimensional matrix, where mf(x,y) is the
#      probability that y improving mutations
#      exist at distance x of p.
# tm:  unidimensional vector, where tm(x) is the
#      size of the basin of attraction at
#      distance x.
# d:   current distance.
# mv:  number of neighbors to be explored.
# w:   weight of the current branch

   mf' = mf; tm' = tm       # initialization
   f = Evaluate (p)         # p is evaluated.
   l = GetNeighbors (p, op)
   tm'(d) += w              # p is at distance d.
   nm = 0                   # improving mutations
   nv = 0                   # neighbors to explore
   for i=1..size(l) do,
      f' = Evaluate (l(i))
      if (f' < f)           # maximization
         vec(nv++) = l(i)   # neighbor stored
      else
         nm++               # improving mutation
      end
   end
   mf'(d,nm) += w           # update mf
   if (nv > mv)             # too many branches
      vec = RandomSort(vec)
      nh = floor(mv+rand(.5))
      w = w*nv/nh           # adjust weight
   else
      nh = nv
   end
   for i=1..nh do,
      [mf', tm']= Explore (vec(i), op, mf', tm',
                           d+1, mv, w)
   end
```

In the SRHC algorithm, the `GetNeighbors` function returns the elements $\mathtt{p}(i)$ such that $\mathtt{op}(\mathtt{p}(i)) = \mathtt{p}$. Subsequenty, the algorithm reduces to `Explore (p, op, mf, tm, 0, mv, 1.0)`, where `mf` and `tm` are initially sparse zero matrices, and `mv` is the average number of neighbors to explore. After executing this algorithm, and assuming *statistical isotropy* [Wei90]), `tm` allows determining the relative sizes of the subsets of the basin of attraction

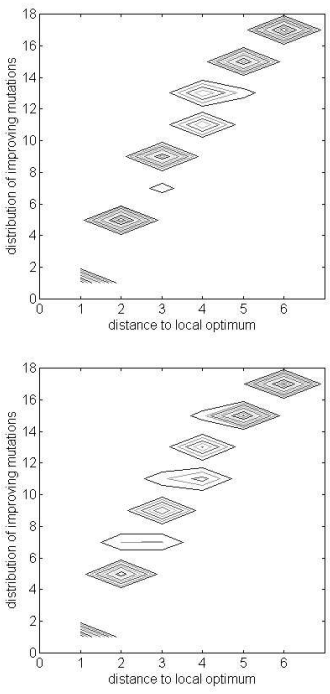

Figure 4: Comparison of the probability distributions for $P(\mathcal{M}_i/\mathcal{D}_n)$ as calculated by exhaustive reverse hillclimbing (top) and stochastic reverse hillclimbing (bottom) on a 3-MMDP (`mv = 1.05`).

of `p` located at distance i ($P(\mathcal{D}_i)$). To be precise, these relative sizes are

$$P(\mathcal{D}_i) = \frac{\mathtt{tm(i)}}{\sum_{j=0}^{\mathcal{D}_{max}} \mathtt{tm}(j)} \tag{17}$$

On the other hand (and under the same assumption), it is possible to calculate the probability distribution $P(\mathcal{M}_i/\mathcal{D}_j)$ from `mf` as follows:

$$P(\mathcal{M}_i/\mathcal{D}_j) = \frac{\mathtt{mf}(i,j)}{\sum_{k=0}^{M_{max}} \mathtt{mf}(k,j)} \tag{18}$$

Now, equation (15) can be rewritten as

$$P(\mathcal{M}_i/\mathcal{I}_n) \cong \sum_{j=0}^{\mathcal{D}_{max}-n} P(\mathcal{D}_j/\mathcal{I}_n) \cdot P(\mathcal{M}_i/\mathcal{D}_j) = \tag{19}$$

$$= \frac{\sum_{j=0}^{\mathcal{D}_{max}-n} P(\mathcal{D}_{j+n}) \cdot P(\mathcal{M}_i/\mathcal{D}_j)}{\sum_{j=0}^{\mathcal{D}_{max}-n} P(\mathcal{D}_{j+n})} \tag{20}$$

The above equation is based on the fact that the probability of falling in $\mathcal{D}_j$ given n improving steps is given by the probability of starting at $\mathcal{D}_{j+n}$, i.e., $P(\mathcal{D}_{j+n})$ (appropriately normalized). As it can be seen in Fig. 4 and 5, the stochastic exploration provides a good approximation of the real probability distribution.

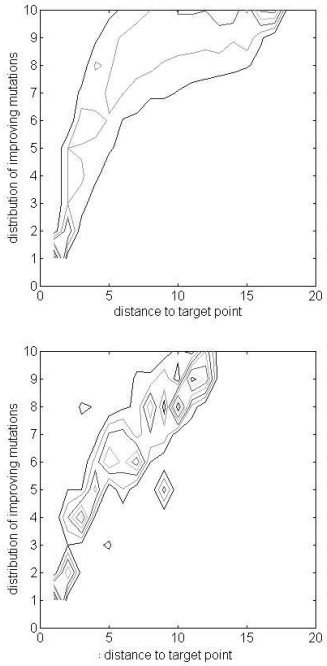

Figure 5: Comparison of the probability distributions for $P(\mathcal{M}_i/\mathcal{D}_n)$ as calculated by exhaustive reverse hillclimbing (top) and stochastic reverse hillclimbing (bottom) on a 10-city TSP (`mv = 1.2`).

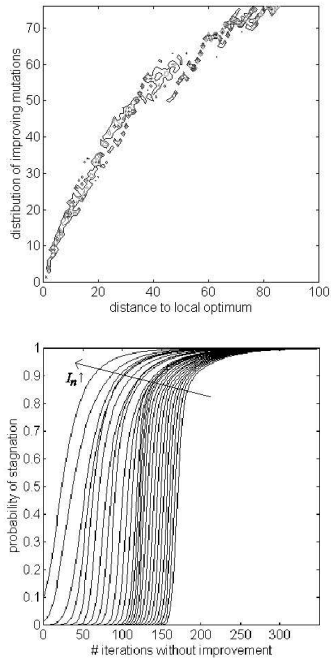

Figure 6: Probability distribution of $P(\mathcal{M}_i/\mathcal{D}_n)$ calculated by stochastic reverse hillclimbing (top) and stagnation probabilities for different values of $\mathcal{I}_n$ (bottom). Both graphics correspond to the `pr76` instance of the Traveling Salesman Problem (taken from the TSPLIB).

This dynamic estimation is not only more accurate, but it has an additional advantage: all information about the target function is empirically obtained and hence no analytical knowledge is needed. This implies that it can be used even for problems whose fitness function is not available in closed form.

As an example of the application of this estimation, consider a (1+10)-LS algorithm applied to the *Traveling Salesman Problem*. This algorithm uses the *neighbor-exchange* operator, i.e., it selects two neighboring positions and swaps their contents. First, the probability distribution $P(\mathcal{M}_i/\mathcal{D}_n)$ is calculated. The result is shown in Fig. 6 (top) for a 76-city problem. According to this distribution, equation (19) is used to determine the family of functions $P(\mathcal{M}_0/E_r, \mathcal{I}_n)$, yielding the curves depicted in Fig. 6 (bottom).

4 Experimental Results

Some experiments have been done in order to asess the quality of these estimations with respect to some other common termination criteria. These criteria are based on extrapolating the behavior of the first run (which is stopped after a high enough number of iterations) to subsequents experiments. To be precise, two parameters are extracted from this first run: the number of iterations which were performed before finding the best solution and the maximum number of successive iterations without change prior to find the best solution. The terms `iter` and `plateau` will be used to denote the criteria that use each of these two parameters respectively. These criteria are compared with the first static estimation (`C#1`) and the dynamic estimation (`C#2`). More precisely, runs are terminated when the probability of stagnation reaches a threshold value ($p \in \{.75, .90, .99\}$ in our experiments).

The experiments have been done utilizing a (1+10)-LS algorithm, using the neighbor-exchange operator and applied to the `pr76` problem. Each termination criterion has been tested ten times and two complementary parameters have been analyzed: the quality of the final solution (compared with a very long run) and the total number of iterations. The results are shown in Fig. 7 and 8.

As it can be seen, `C#1` and `C#2` offer the best results. Notice that `C#1` is successful at not stopping the algorithm before stagnation, but it requires a high number of iterations. On the contrary, both `plateau` and `iter` stop the algorithm after a low number of iterations, but the quality of the results is worse. The dynamic criterion `C#2` offers the best tradeoff between these two measures. The algorithm is almost optimally stopped after a much lower number of iterations than `C#1`.

5 Conclusions and Future Work

This work has studied the detection of stagnation states in iterated local search algorithms. Different estimations have been presented for that purpose. The first one is very easily calculated and can be directly applied to any problem, thus constituting a very good starting point as a termination criterion. However, we hypothesize that, as the number of possible mutations increases, this estimation tends to be more pesimistic. This is grounded in the higher weight assigned to configurations with many acceptable mutations. Notice that, as the algorithm evolves, the search is focused in configurations with a low number of such acceptable mutations. For that reason, a dynamical criterion is proposed in order to take this effect into account. This dynamical criterion considers both the features of the fitness landscape induced by the algorithm and the evolution of the algorithm. For the former aspect, the structure of the search space is studied using a stochastic version of the reverse hillclimbing algorithm. This information is used to estimate the probability distribution of different configurations as a function of the history of the algorithm.

The experimental results are satisfactory. The dynamical estimation provides a good tradeoff between the number of iterations and the quality of the results. Moreover, this tradeoff is tuneable by adjusting the threshold probability p used to determine the termination of the algorithm.

Future work could be directed to improve this dynamical estimation using more problem information, e.g., studying the fitness correlation with respect to the number of succesful mutations. This information could be further exploited considering more aspects of the evolution of the algorithm, e.g., the distribution of non-improving mutations.

Bibliography

[BT95] T. Bickle and L. Thiele. A mathematical analysis of tournament selection. In L. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 9–16, San Francisco, CA, 1995. Morgan Kaufmann.

[Gol89] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading, MA, 1989.

[Gol95] D.E. Goldberg. Critical deme size for serial and parallel genetic algorithms. Technical Report #95002, IlliGAL, 1995.

[Jon95] T.C. Jones. *Evolutionary Algorithms, Fitness Landscapes and Search.* PhD thesis, University of New Mexico, 1995.

[JR93] T.C. Jones and G.J.E. Rawlins. Reverse hillclimbing, genetic algorithms and the busy beaver problem. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 70–75, San Mateo, CA, 1993. Morgan Kaufmann.

[Wei90] E.D. Weinberger. Correlated and uncorrelated fitness landscapes and how to tell the difference. *Biological Cybernetics*, 63:325–336, 1990.

[WGM94] D. Whitley, S. Gordon, and K. Mathias. Lamarckian evolution, the baldwin effect and function optimization. In Y. Davidor, H.-P. Schwefel, and R. Männer, editors, *Parallel Problem Solving From Nature - PPSN III*, pages 6–15. Springer-Verlag, 1994.

[WM95] D.H. Wolpert and W.G. Macready. No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe Institute, 1995.

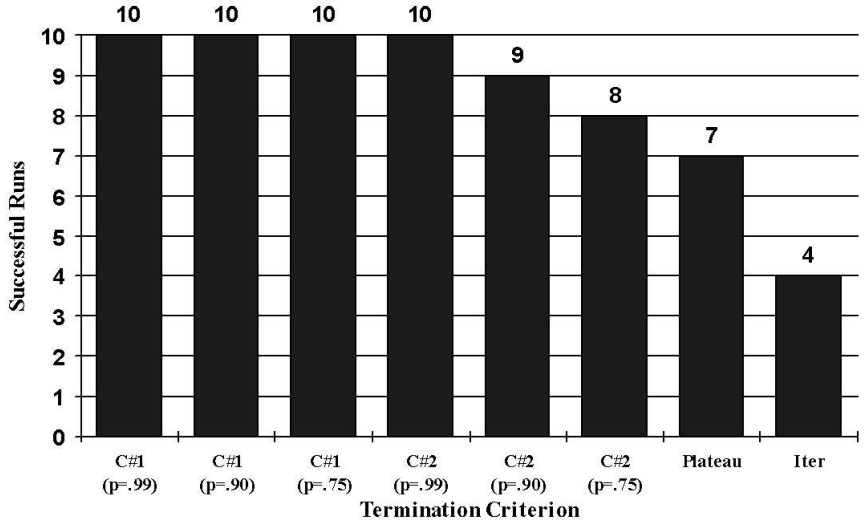

Figure 7: Number of runs in which stagnation is succesfully detected.

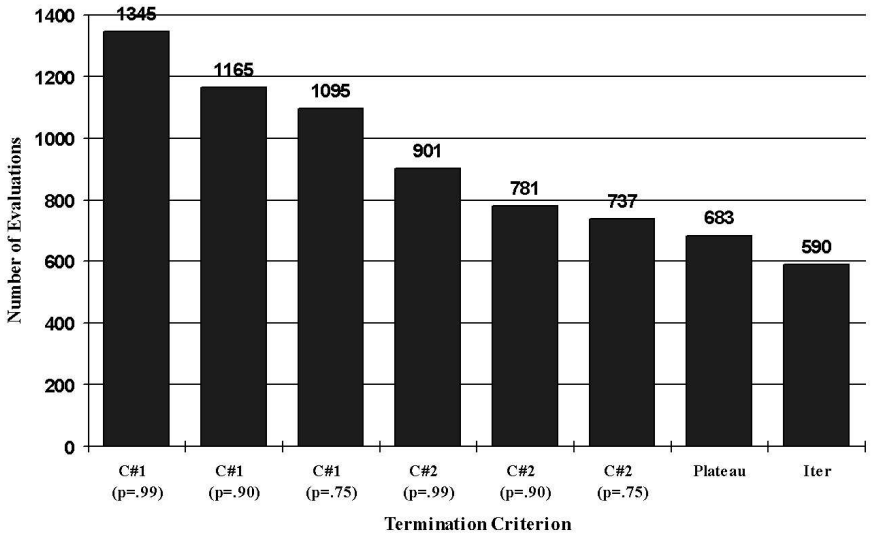

Figure 8: Average number of iterations before termination.