Study of Neighborhood search operators for unitation functions

In this paper we study the behavior of neighborhood search algorithms in optimization of unitation functions. The influence of two neighborhood search strategies is analyzed. The expected number of steps required by these algorithms to reach the optimum is derived. The analytical results achieved correspond to previous simulations.

1 Introduction

Unitation functions are functions defined in the finite n-dimensional binary space whose values are related to the number of components set to 1 (see section 2 for a formal definition) and have a number of attributes that make them particularly appealing for the investigation of different optimization algorithms. These functions have received a particular attention in Genetic Algorithms (GAs) [1] and Estimation Distribution Algorithms (EDAs) [6], two Population Based Search Methods that use Selection (PBSMS), commonly used as optimization methods. The performance of GAs and EDAs for unitation functions has been extensively investigated.

Unitation functions are also important because they allow the study of the behavior of optimization algorithms in the presence of multiple local and global optima. They are useful to understand, for example, how EDAs optimize functions by transforming the original landscape in the landscape given by the average fitness of the population, a result that was presented in [5].

One important question that arises is how single searchers behave for these functions. In [4] the computational complexity of Simulated Annealing (SA) [3] for a fixed temperature and neighborhood sizes was investigated in the framework of the optimization of unitation functions. A theoretical comparison of stochastic local search using large neighborhoods with a local search using optimal temperature schedules was done. In [7] a comparison between EDAs and some local searchers for some unitation problems was presented.

In this paper we analyze the influence of the way the neighborhood is defined in the performance of neighborhood based local searchers. We derive a formula for estimating the probability of reaching the optimum in one single step of the search for a subclass of unitation functions. Results are applied to the calculation of the number steps needed by the local searcher to reach the optimum in the case of unitation functions with gaps. The outline of the paper is as follows: Next section introduces the class of unitation functions. Section 3 presents a neighborhood based search algorithm based on the Boltzmann distribution. The analysis of two types of neighborhood search operators is developed in section 4. Section 5 shows how the derived results can be used to estimate the number of steps to reach the optimum for one particular function, the results on the approximation are validated comparing with previous simulations. We present our conclusions in section 6.

2 Unitation functions

Let $X = (X_1, \ldots, X_n)$ be a tuple of random variables and $X \in B^n$ where B^n is the finite n-dimensional binary space. We will use x to denote a value of X, and x_i to denote a value of X_i, the i-th component of X. Let f be a function such that $f(x) : B^n \mapsto R^{\geq 0}$, and let $u(x) = \sum_{i=1}^{n} x_i$.

Definition 1. *$f(x)$ is a unitation function if* $\forall x, y \in B^n, u(x) = u(y) \Rightarrow f(x) = f(y)$

$u(x)$ is itself a unitation function, usually called *Onemax* because it reaches the maximum at $x = (1, \ldots, 1)$. A unitation function can be defined in terms of its unitation value $u(x)$ or, in a simpler way, u. One example is the *Jump* function (1) [4]. The parameter gap ($gap \in N, 0 \leq gap < n$) of this function defines the number of steps one has to go downhill in order to reach the unique maximum. For $gap = 0$ we have the very simple *Onemax* function. As the parameter gap increases, so does the difficulty of the function. The graphic of function *Jump* is shown in figure 1.

$$Jump(n, gap, u) = \begin{cases} u & u < n - gap \\ 2 * (n - gap - 1) - u & n - gap \leq u \\ n & u = n \end{cases} \tag{1}$$

3 Neighborhood based search algorithms

Neighborhood based search algorithms are local search methods that start from a single point, and proceed the search for the optimum making transitions to points in a predefined neighborhood of the current one. The ways in which a neighborhood is defined, the probabilities of visiting the different of visiting the different points in the neighborhood, and the criteria to accept transitions to the neighbors determine the dynamics of the search. We will consider neighborhood search algorithms defined on B^n. For this space the size of the neighborhood is defined as the number of solutions that belong to the neighborhood.

Some neighborhood search methods use a dynamical variable to determine transitions in the space of solutions. This is the case Simulated Annealing (SA) which uses the temperature as a dynamical variable that changes with time, and may allow the system

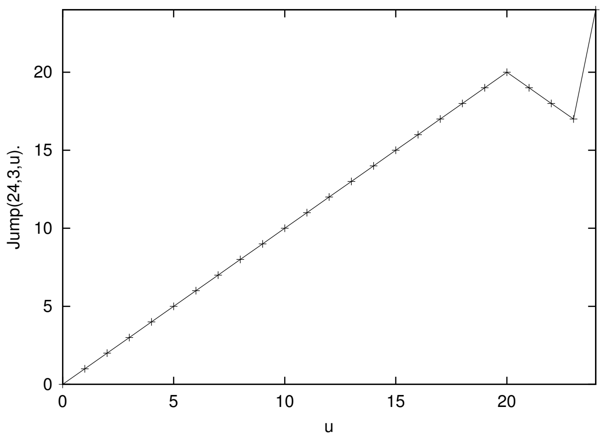

Figure 1: Function Jump, $n = 24$, $gap = 3$.

to make transitions which would be improbable at a fixed temperature. In SA the neighborhood of a point is usually comprises the set of points that are in an 1-bit distance from the current point.

Recently Mühlenbein and Zimmermann [4] have shown that for stochastic local search algorithms like SA the size of the neighborhood is more important than the temperature schedule. In fact, in the field of local optimization, it has been reported that updating more than one variable at the time can be a good heuristic for escaping local optima when sequential optimization algorithms are used. This is the case for example of GSAT [2], a very effective local optimization strategy used to solve the Satisfiability problem, where the number of variables to be updated in each step is not fixed. We will constrain our analysis to the case of the algorithms that use neighborhood search, where the neighborhood size s is fixed. A transition from the current state to the next state is done by changing at most s variables of the current state. The search can be done in two steps.

- The s variables that will be changed are selected.
- The values of all or some of the variables are changed.

Let Ng represent a set of variables to be updated, s is the number of variables in Ng, and we will refer to the variables that are not in Ng as X/Ng. For $s = n$ we have $X/Ng = \emptyset$. x_{Ng} is the sub-vector of the vector x formed by variables in Ng. The close neighborhood $v_{Ng}(x)$ of x includes x and the set of points that can be accessed by changing the values in x_{Ng}. From now on we understand a neighborhood as a closed neighborhood.

In the neighborhood search algorithm we use, the set Ng of s variables is uniformly selected from X without replacement. Available information may be use to select the neighbors in a "convenient" biased way. Results for the case where the structure of the function is used in the selection of the blocks can be found in [7]. The new configuration of variables in Ng is found sampling from a neighborhood probability P that is an input of the algorithm. P is defined in the space of the 2^s binary configurations, and it is fixed for any set Ng.

The neighborhood search algorithm is shown in algorithm 1. The algorithm receives as a parameter the block size s, that can be also understood as the maximum number of variables that will change their values together.

Algorithm 1: **Neighborhood based searcher**

```
1  Set $t \Leftarrow 0$. Generate a random initial point $x^0$.
2  do {
3      Select a set $Ng$ of $s$ different uniformly selected variables of $X$.
4      Propose a new point $x'$ such that $x'_i = x_i$ if $X_i \in X/Ng$ and $x'_i$ is sampled
       from the neighborhood probability $P$ for $X_i \in X/Ng$
5      if $f(x') \geq f(x)$ then $x^{t+1} = x$.
6      $t \Leftarrow t + 1$.
7  } until Termination criteria are fulfilled
```

We have used two different ways of defining transition probabilities. These two ways actually define different types of neighborhoods, and we will refer to them as $Neig1$ and $Neig2$. In $Neig1$ a uniformly random value x'_{Ng} is selected among the $2^s - 1$ possible values. $Neig2$ has been implemented as in [4], the probability is uniform in the space of the number of the variables that can change their value together. Table 3 shows an example of how the neighborhood $Neig1$ and $Neig2$ are constructed for a point x. Note that while P_{Neig1} is uniform in the 2^s neighbors, P_{Neig2} is not, however P_{Neig2} is uniform in the space of the unitation. Finally if the value of the function at the proposed point x' is not worse that at the current one the transition is made.

$Ng = \{1,3,5\}$	$v_{\{1,3,5\}}(x)$							
$x = (00000)$	00000	10000	00100	00001	10100	10001	00101	10101
u	0	1			2			3
$P_{Neig1}(x)$	0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125
$\sum_x P_{Neig1}(x)$	0.125	0.375			0.375			0.125
$P_{Neig2}(x)$	0.250	0.083	0.083	0.083	0.083	0.083	0.083	0.250
$\sum_x P_{Neig2}(x)$	0.250	0.250			0.250			0.250

Table 1: Definition of two different neighborhood transition probabilities for $v_{1,3,5}(00000)$.

4 Analysis of the neighborhood operators

We consider again a maximization problem. Let us suppose that the optimum of $f(x)$ is located at point $x = (1, 1, \ldots, 1)$. We want to estimate the probability of making a transition from a point with unitation u to the optimum using a neighborhood search algorithm with neighborhood size s. Let us call this probability P_{trans}. To hit the optimum the following facts have to occur.

- The $(n-u)$ variables with value 0 are selected among the s variables.
- The new proposal changes the values of these $(n-u)$ variables and keep the values of the rest of the variables intact.
- The new proposal is accepted.

Then P_{trans} can be factorized as:

$$P_{trans} = P_{sel} \cdot P_{opt} \cdot P_{acceptance}$$

$$P_{sel} = \frac{\binom{u}{s-(n-u)}}{\binom{n}{s}} \tag{2}$$

Where P_{sel} is the probability of having the $(n-u)$ variables among the s variables selected. P_{opt} is the probability of changing the $(n-u)$ variables to 1, while keeping the remaining $s-(n-u)$ in their current values. P_{opt} depends on the way the neighborhood structure is defined. Finally, given that x' is the new proposal, x' will be accepted if $f(x') \geq f(x)$ Thus, the probability of hitting the optimum is equal to the probability of selecting the optimum as the next proposal. If the current solution has unitation u, and $n-u > s$, the probability of generating the optimum as the new proposal is 0 because in this case it is impossible to make the transition in just one step. It could be the case that the $n-u$ variables that need to be changed are among the set of s variables selected. In this case the transition to the optimum will be done, i.e. $P_{opt} = 1$. The analysis leads to the following theorem.

Theorem 1. *The probability that a random neighborhood based search algorithm with neighborhood size s and that always accept better points reaches the optimum in one step is given by:*

$$P_{trans} = \frac{\binom{u}{s-(n-u)}}{\binom{n}{s}} \cdot P_{opt} \tag{3}$$

Moreover, equation (2) *gives the maximum probability of reaching the optimum in one step.*

Now let us consider P_{opt} for $Neig1$ and $Neig2$ introduced in the previous section. In $Neig1$ an assignment for the s variables is uniformly random generated in the space of neighbors. This is:

$$P_{trans}^{Neig1} = \frac{\binom{u}{s-(n-u)}}{\binom{n}{s}} \cdot \frac{1}{2^s} \tag{4}$$

This would correspond to a search algorithm with uniform transition rules. But uniform transition rules do not imply a uniform search of the space. For the particular case of uniform search, the $Neig2$ case, the probability of selecting a new assignment for the s variables must satisfy that all the points of the neighborhood are visited with the same probability.

Let us consider the probability of having a neighbor y of x whose Hamiltonian distance from x is h (i.e. $H(x,y) = h$). Obviously, the probability for $h > s$ is zero. For $h \leq s$ this probability is:

$$P_{H(x,y)=h} = \frac{\binom{n}{h}}{\sum_{h'=0}^{s} \binom{n}{h'}} \tag{5}$$

Then P_{opt}^{Neig2} has to be calculated according to the distance to the optimum h_{opt}. If the current solution has unitation u then $h_{opt} = n - u$ and

$$P_{opt}^{Neig2} = \frac{\binom{n}{n-u}}{\sum_{h'=0}^{s} \binom{n}{h'}} \cdot \frac{1}{\binom{s}{n-u}}$$

where the first term in the expression corresponds to the probability of changing exactly $n-u$ components while the second expression is the probability of finding the right $n-u$ variables among the s variables selected.

Theorem 2. *The probability that a random neighborhood based search algorithm $Neig2$ with neighborhood size equal s reaches the optimum in one step is:*

$$P_{trans}^{Neig2} = \frac{1}{\sum_{h'=0}^{s} \binom{n}{h'}} \tag{6}$$

Proof: Substituting (6) in (3) we get:

$$P_{trans}^{Neig2} = \frac{\binom{u}{s-(n-u)}}{\binom{n}{s}} \cdot \frac{\binom{n}{n-u}}{\sum_{h'=0}^{s} \binom{n}{h'}} \cdot \frac{1}{\binom{s}{n-u}}$$

Considering the case when $s = n - u + a$, $a \geq 0$ and substituting in (7) we arrive to:

$$P_{trans}^{Neig2} = \frac{\binom{u}{a}}{\binom{n}{n-u+a}} \cdot \frac{\binom{n}{n-u}}{\sum_{h'=0}^{s} \binom{n}{h'}} \cdot \frac{1}{\binom{n-u+a}{n-u}}$$

$$= \frac{u!}{(u-a)!a!} \cdot \frac{(n-u+a)!(u-a)!}{n!} \cdot \frac{n!}{(n-u)!u!)} \cdot \frac{(n-u)!a!}{(n-u+a)!} \cdot \frac{1}{\sum_{h'=0}^{n-u+a} \binom{n}{h'}}$$

$$= \frac{1}{\sum_{h'=0}^{n-u+a} \binom{n}{h'}} \tag{7}$$

Finally, substituting $a = s - n + u$ in (7) we obtain the expression (6).

Corollary 1. *P_{trans}^{Neig2} is maximal when $s = n - u$.*

5 Structure of the neighborhood, $Jump$ function

We investigate now the role of the neighborhood structure in the case of the $Jump$ function. For this function most of the steps spent by a neighborhood search algorithm with neighborhood size s are used to pass from a local optimum with unitation $n-gap-1$ to the optimum of unitation n.

The expected number of steps to reach the optimum can be calculated as the inverse of the probability for reaching it. Using (4), and (7) we estimate the total number of steps to reach the optimum of the $Jump$ function with gap equal gap when $Neig1$ and $Neig2$ are used. We substitute u by $n - gap - 1$ and s by $gap + 1 + a$.

$$N_{Neig1} = \frac{\binom{n}{gap+1+a}}{\binom{n-gap-1}{a}} \cdot 2^{gap+1+a} \tag{8}$$

$$N_{Neig2} = \sum_{h'=0}^{gap+1+a} \binom{n}{h'} \tag{9}$$

For the $Jump$ function it is clear that the minimal number of steps to reach the optimum needed by the $Neig2$ (equation (9)) is achieved when $s = gap + 1$, otherwise the number of steps is incremented. This result demonstrates the following theorem presented as an empirical law in [4].

n	g	τ	N_{Neig2}	g	τ	N_{Neig2}	g	τ	N_{Neig2}
8	1	47.6	37.0	2	96.7	93.0	3	164.6	163.0
16	1	191.0	137.0	2	718.6	697.0	3	2538.5	2517.0
24	1	430.0	301.0	2	2379.1	2325.0	3	13026.6	12951.0
32	1	764.6	529.0	2	5590.5	5489.0	3	41633.0	41449.0
64	1	3059.4	2081.0	2	44202.2	43745.0	3	680863.5	679121.0

Table 2: Comparison of the simulation results τ for the function $Jump$ with the expected number of steps N_{Neig2}

The expected number of steps N_{Neig2} for the $Jump$ function and different number of variables are presented in table 2. The predictions are compared with results of the simulations appeared in [4]. As N_{Neig2} is just the number of the steps required to jump from the local optimum to the optimum, it is only a lower bound of the expected passage time τ. Nevertheless it can be appreciated in the table how close is the prediction.

To analyze the case of $Neig1$ we consider a simplification of equation (8) for N_{Neig1}.

$$N_{Neig1} = \frac{n!a!(n-(gap+1+a))!}{(n-(gap+1+a))!(gap+1+a)!(n-gap-1)!} 2^{gap+1+a}$$

$$= \frac{n!a!}{(gap+1+a)!(n-gap-1)!} 2^{gap+1+a}$$

$$= \frac{a!(n-gap-1)!\prod_{i=1}^{gap+1}(n-gap-1+i)}{(n-gap-1)!a!\prod_{i=1}^{gap+1}(a+i)} 2^{gap+1+a}$$

$$= \left(\prod_{i=1}^{gap+1} \frac{n-gap-1+i}{a+i}\right) 2^{gap+1+a} \qquad (10)$$

In (10) it can be seen that the number of steps depends on two terms. When a is increased the first term decreases but the second one gets exponentially higher. When $a = n - gap - 1$ the first term is 1 and N_{Neig1} is equal to the size of the space. If $a = 0$ then the number of steps becomes:

$$N_{Neig1} = \binom{n}{gap+1} 2^{gap+1}$$

and this value can be even higher than 2^n. So, for the $Jump$ function the $Neig2$ makes a more efficient search. Figure 2 shows how N_{Neig1} and N_{Neig2} scale when the number of variables is increased for the $Jump$ function, $gap = 1, 5$. The size of neighborhood for $Neig1$ and $Neig2$ are respectively $(2gap + 1)$ and $(gap + 1)$. In the figure the y axes is log-scaled, vertical lines show the first n for which the computation of the number of steps is possible $(n > s)$. It can be seen in the figure that the number of steps is always higher for $Neig2$. The difference between the number of steps needed by both algorithms can be intuitively appreciated if we realize that the average number of variables that change their value in every step of the $Neig1$ is less than the average for $Neig2$. As a consequence the first algorithm needs more steps for finding a way to cross the gap.

6 Conclusions

For unitation functions we have calculated which is the maximum probability for a random neighborhood based search algorithm with neighborhood size s to reach the optimum in one step. This formula allows to estimate the average number of steps needed by the algorithm to jump from a local suboptimum of unitation u to the optimum. We have also presented the formulae for calculating this probability for two commonly used neighborhood structures, $Neig1$ and $Neig2$. In the case of $Neig2$ we have also shown that the optimal choice of the neighborhood size is $s = n - u$. Results have been applied to derive the number of steps needed by function $Jump$ to reach the optimum, demonstrating the conjecture presented in [4]. Concerning the differences between $Neig1$ and $Neig2$ algorithms, an important conclusion of our analysis is that the way transition probabilities of the neighborhood are defined is as critical for the efficiency of the search as the own choice of the neighborhood size is.

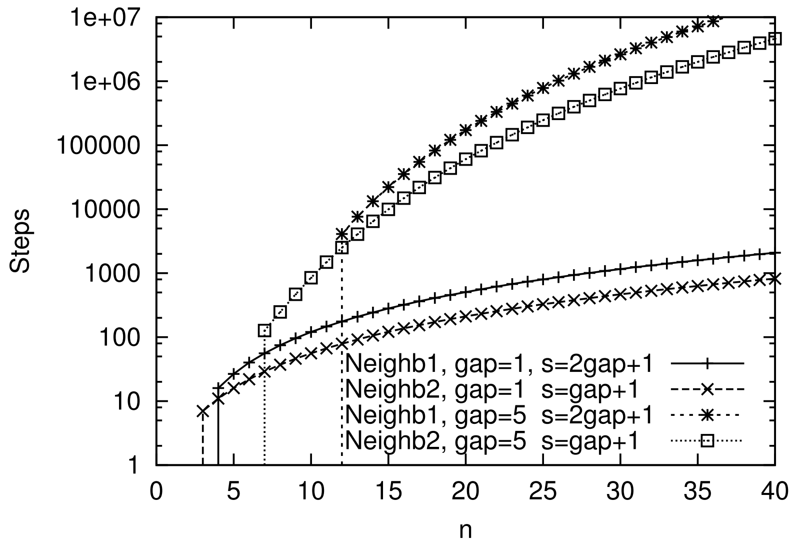

Figure 2: Expected number of steps of the random blocked Gibbs Sampling for the *Jump* function when n is increased.

7 Acknowledgments

The author thanks to Heinz Muehlenbein for having introduced him to the problem and to Li-Vang Lozada Chang for his comments on the paper, valuable discussion and insight.

References

[1] J. H. Holland. *Adaptation in natural and artificial systems.* University of Michigan Press, Ann Arbor, MI, 1975.

[2] K. Kask and R. Dechter. GSAT and local consistency. In *Proceedings of the 14th IJCAI*, pages 616–622, Montreal, Canada, 1995.

[3] S. Kirkpatrick, C. D. J. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, May 1983.

[4] H. Müehlenbein and J. Zimmermann. Size of neighborhood more important than temperature for stochastic local search. In *Proceedings of the 2000 Congress on Evolutionary Computation CEC00*, pages 1017–1024. IEEE Press, 2000.

[5] H. Mühlenbein and T. Mahnig. Evolutionary computation and beyond. In Y. Uesaka, P. Kanerva, and H. Asoh, editors, *Foundations of Real-World Intelligence*, pages 123–188. CSLI Publications, Stanford, California, 2001.

[6] H. Mühlenbein and G. Paaß. From recombination of genes to the estimation of distributions I. Binary parameters. In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN IV*, pages 178–187, Berlin, 1996. Springer Verlag. LNCS 1141.

[7] R. Santana and H. Mühlenbein. Blocked stochastic sampling versus Estimation of Distribution Algorithms. In *Proceedings of the 2002 Congress on Evolutionary Computation*, volume 2, pages 1390–1395. IEEE press, 2002.