The Road to VEGAS: Guiding the Search over Neutral Networks

Marie-Éléonore Marmion
Université Lille 1
LIFL – CNRS – INRIA
marie-eleonore.marmion@lifl.fr

Clarisse Dhaenens
Université Lille 1
LIFL – CNRS – INRIA
clarisse.dhaenens@lifl.fr

Laetitia Jourdan
Université Lille 1
LIFL – CNRS – INRIA
laetitia.jourdan@inria.fr

Arnaud Liefooghe
Université Lille 1
LIFL – CNRS – INRIA
arnaud.liefooghe@lifl.fr

Sébastien Verel
Univ. Nice Sophia-Antipolis
INRIA
sebastien.verel@inria.fr

ABSTRACT

VEGAS (Varying Evolvability-Guided Adaptive Search) is a new methodology proposed to deal with the neutrality property that frequently appears on combinatorial optimization problems. Its main feature is to consider the whole evaluated solutions of a neutral network rather than the last accepted solution. Moreover, VEGAS is designed to escape from plateaus based on the evolvability of solutions, and on a multi-armed bandit by selecting the more promising solution from the neutral network. Experiments are conducted on NK-landscapes with neutrality. Results show the importance of considering the whole identified solutions from the neutral network and of guiding the search explicitly. The impact of the level of neutrality and of the exploration-exploitation trade-off are deeply analyzed.

Categories and Subject Descriptors

I.2.8 [**Computing Methodologies**]: Artificial Intelligence—*Problem Solving, Control Methods, and Search*

General Terms

Algorithms

Keywords

Neutrality, Local search, Evolvability, Adaptive search, Multi-Armed Bandit

1. MOTIVATIONS

Due to their ability to find satisfying solutions with high efficiency and effectiveness, the design of local search approaches is still a prominent issue for hard combinatorial optimization. This class of methods is also of strong interest since they are generally quite simple to implement. However, in some circumstances, these methods may have more difficulties. One of the most critical situation arises when the problem under study holds neutrality, which is the case for many problems from combinatorial optimization, like scheduling or satisfiability. This property means that a lot of different solutions share the same fitness value. In such a case, natural questions hold: Once the neutrality of a problem is known, how could the search exploit it? How could the search be guided to exploit this neutrality with success? The aim of this article is to propose an efficient approach, based on simple local search principles and adaptive mechanisms, that exploits the inherent neutrality to a problem, without inhibiting too much the search for neutrality-free problems. Therefore, an experimental comparison will be driven between approaches that do or do not exploit the neutrality explicitly during the search process.

The central idea of the article is based on two observations. On the one hand, up to now, the neutrality property has been under-exploited on the design of search algorithms, with the exceptions of a few examples that will be detailed later in the paper. On the other hand, adaptive search, based on a multi-armed bandit framework, has been successfully applied to parameter control in the context of evolutionary algorithms, in particular in adaptive operator selection [7]. In this paper, the goal is not to adapt parameters along the search. Instead, we propose an original adaptive algorithm, based on a multi-armed bandit framework, to guide the search over neutral networks. When the search is stuck on a plateau, VEGAS adaptively selects the more-promising solution whose neighborhood has to be explored. Hence, two main questions are addressed in this paper:

(i) How to consider neutrality, and then neutral networks, during the search process? How to avoid an arbitrary choice to be made during a neutral search?

(ii) How to guide the search adaptively over neutral networks in order to explore the neighborhood of a solution that is more likely to escape a plateau by the top?

The paper is organized as follows. Section 2 introduces fundamental definitions together with previous works on neutrality-based and adaptive search methods. The proposed VEGAS algorithm, where the search is adaptively guided by the evolvability of solutions, is introduced in Section 3. Section 4 presents experimental results on the performance of VEGAS, on the influence of its single problem-independent parameter and on the influence of the degree of neutrality of the problem at hand. The last section gives a summary of results and discusses future works.

2. BACKGROUND

This section introduces the main concepts dealing with landscape analysis and the design of neutrality-based and adaptive search methods.

2.1 Local Search, Neutrality and Evolvability

A fitness landscape [18] is a triplet $(S, \mathcal{N}, f)$ where S is a set of admissible solutions (*i.e.* the search space), $\mathcal{N} : S \longrightarrow 2^S$ is a neighborhood structure, and $f : S \longrightarrow \mathbb{R}$ is a fitness function that can be pictured as the *height* of the corresponding solutions, here assumed to be maximized. A *neighborhood structure* is a mapping function that assigns a set of solutions $\mathcal{N}(s) \subset S$ to any feasible solution $s \in S$. $\mathcal{N}(s)$ is called the *neighborhood* of s, and a solution $s' \in \mathcal{N}(s)$ is called a *neighbor* of s. We then may define a *local optimum* as: solution s^* is a *local optimum* iff no neighbor has a better fitness value: $\forall s \in \mathcal{N}(s^*)$, $f(s^*) \geq f(s)$.

The importance of *selective neutrality* as a significant factor in evolution was stressed by Kimura [12] in the context of evolutionary theory. The relevance and benefits of neutrality for the robustness and evolvability in living systems have been recently discussed by Wagner [23]. There is a growing evidence that such large-scale neutrality is also present in artificial landscapes. Not only in combinatorial fitness landscapes such as randomly generated SAT instances [8], cellular automata rules [22] and many others, but also in complex real-world design and engineering applications such as evolutionary robotics [17, 16], evolvable hardware [10, 21], genetic programming [2, 24] and scheduling [14].

A *neutral neighbor* of a given solution $s \in S$ is a neighboring solution $s' \in \mathcal{N}(s)$ with the same fitness value: $f(s') = f(s)$. Given a solution $s \in S$, the set of neutral solutions in its neighborhood is defined by $\mathcal{N}_n(s) = \{s' \in \mathcal{N}(s) \mid f(s') = f(s)\}$. The *neutral degree* of a solution is the number of its neutral neighbors. A fitness landscape is said to be *neutral* if there are many solutions with a high neutral degree.

A *neutral network*, or *plateau*, denoted here by NN, is a connected sub-graph of solutions with respect to neighborhood relation whose vertices are solutions with the same fitness value. There exists an edge between solutions s and s' when s' is a neutral neighbor of s. A *portal* in a NN is a solution that has at least one neighbor with a better fitness value, *i.e.* a greater fitness value in a maximization context.

2.2 Neutrality-based Search

When dealing with neutrality, two extreme local search approaches are usually designed. The first one simply ignores neutrality. One of the simplest algorithm is the first-improvement hill-climbing (FIHC), where the first evaluated neighbor that *strictly* improves the current fitness value is accepted. In other words, the heuristic does not move on a neutral neighboring solution, and prefers to keep exploring the neighborhood of the current solution, assuming that the neutral neighbor will not lead to better solutions. At the opposite, some local search approaches always accept the first visited neighbor with the same fitness value. A typical example is the *Netcrawler* process [4], that consists of a random neutral walk with a mutation mode adapted to local neutrality. The per-sequence mutation rate is optimized to jump from one NN to another. Stewart [19] also proposed an *Extrema Selection* for evolutionary optimization in order to move on promising solutions in a neutral search space. The selection aims at accelerating the evolution during the search process once most solutions from the population have reached the same level of performance. To each solution is assigned an endogenous performance during the selection step to explore the search space area with the same performance more largely, with the assumption that it will help to reach solutions with better fitness values. Additionally, the NILS (Neutrality-based Iterated Local Search) algorithm, recently proposed in [13], shows interesting results and enhances the interest of taking neutrality into account for flowshop scheduling problems.

All those heuristics focus the search on the last solution found along the NN. Their bet is that the new accepted solution has more chance to lead to better solutions than the previous one, because no better solution was (yet) evaluated in its neighborhood. But, when no better neighbor has been found again, the heuristic prefers to move to a new solution (in another part of the search space) than to go back on a previously accepted solution, even if it could seem more promising *a posteriori*. Hence, nothing motivates the choice of exploring the neighborhood of the last-accepted solution from the NN rather than any other. Most of the time, such a choice appears quite arbitrary. We believe that there might exist a better trade-off between these two extreme cases (ignoring neutrality, and starting with the last-accepted neutral solution), by keeping memory of the solutions evaluated along the NN.

2.3 Adaptive Search

Autonomous self-management search receives more and more attention from the past years due to the increasing complexity of the search methods and problems. The general goal of those methods is to automatically adapt their mechanism to the changing problem conditions. The aim of parameter control is the on-line setting of parameters such as the representation of solution, the stochastic operators (mutation, crossover), the selection operators, the application rate of those operators, etc. [6]. In combinatorial optimization, adaptive methods are often preferred over self-adaptive ones which increase the size of the search space. From the search history, adaptive methods select the new parameter setting. Different rules are used for selection: probability matching, adaptive pursuit [20] which attaches a probability of success to each operator, the multi-armed bandit [5], and so on. The multi-armed bandit framework is a sequential learning model, mostly studied in game theory, dealing with the trade-off between exploration and exploitation. It considers a set of K independent arms, each one having some reward following an unknown distribution. An optimal selection strategy maximizes the cumulative reward along time. The Upper Confidence Bound (UCB) strategy [1], which is asymptotically optimal, has been used in the context of adaptive operator selection [5]. To each arm, which is an operator in that context, is associated an empirical reward which reflects its quality. Then, the operator (arm)

Fitness	N° Op
10	1
9	1
8	2
7	1
7	1
7	2
7	2
5	1
4	1
4	2
4	2
3	1
2	2
1	2

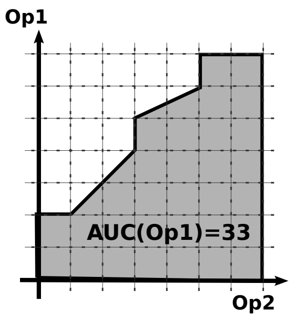


Figure 1: Example of the AUC reward computed for an operator op_1. The list of fitness values produced by each possible operator is sorted in the decreasing order. A curve from the point $(0,0)$ is drawn according to this list: When the operator op_1 is concerned, the curve follows the op_1 axe, otherwise it follows the other axe. The area under this curve is the reward of the operator op_1 [7].

with the best score is selected, where the upper confidence bound of the reward defines this score:

$$\text{score}_{i,t} = \hat{r}_{i,t} + C\sqrt{\frac{\log \sum_k n_{k,t}}{n_{i,t}}} \tag{1}$$

where for the time step t, $\hat{r}_{i,t}$ is the empirical reward of operator i, $n_{i,t}$ is the number of times that the operator i has been tried. C is a problem-independent parameter representing a scaling factor which controls the trade-off between exploitation and exploration.

The measure of the operator quality (reward) has an impact on the efficiency of the method. Different measures of reward have been proposed: the average fitness improvement between parent and offspring, the maximum fitness improvement from a time windows [5], or more recently *the Area Under the Curve* [7]. This credit assignment uses the comparison of solution fitness values produced by each operator. This method uses the rank instead of a normalized fitness improvement. Then, the bandit adaptive technique becomes more invariant to fitness function transformation, and the sensitivity according to the parameter C decreases. The Figure 1 gives an example of AUC computation. The details is not given due to space limitation, the reader is refereed to [7].

3. VEGAS

This section presents the VEGAS (Varying Evolvability-Guided Adaptive Search) algorithm. As seen in the previous section, the First-Improvement Hill-Climbing (FIHC) algorithm and the Netcrawler (NC) algorithm are based on different strategies. The first one does not take neutrality into account, whereas the second one proposes a specific way to deal with it (by always exploring the last accepted - better or neutral - solution). The aim of VEGAS is to take the neutrality explicitly into account and to propose an efficient way to guide the search on NN. First, we reconsider the search over a NN. Second, a guiding strategy, based on the evolvability of solutions, is proposed.

Algorithm 1 VEGAS

```
$S = \{s_0\}$
while $\exists s \in S$ such that $s$ is not visited do
    $s \leftarrow$ select(S)
    Choose a solution $s' \in \mathcal{N}(s)$ at random (no repetition)
    if $f(s') > f(s)$ then
        $S \leftarrow \{s'\}$
    else if $f(s') = f(s)$ then
        $S \leftarrow S \cup \{s'\}$
    end if
    Update rewards($s, s'$)
end while
Return $s \in S$
```

3.1 Considering Neutral Networks

First, let us define some useful terms. A solution is said to be *evaluated* if its fitness value has been computed. A solution is marked as *visited* if its neighborhood has been completely evaluated, otherwise it is *non-visited*. The neighborhood of a solution is explored in a random order without repetition. Let us remind that a NN, also known as *plateau*, is a set of neighboring solutions with the same fitness value. The set of evaluated solutions from the current NN is denoted by S.

Let us consider a simple local search algorithm that iteratively improves a current solution s by exploring its neighborhood. As soon as a strictly improving neighboring solution is found, it is accepted and replaces the current solution. As long as a neutral neighboring solution is evaluated, a particular strategy is applied in order to iteratively build the set S. As opposed to a NC, the main idea of our approach is to consider the whole set of evaluated solutions from the current NN (*i.e.* S) instead of a single one (the last-evaluated solution). Now, the question is: Which solution $s \in S$ to select in order to evaluate a new neighboring solution? For instance, when no particular information is computed, this solution can be selected at random. The algorithm stops once all solutions from S are marked as visited, *i.e.* the neighborhood of all solutions from S has been explored. When there is no neutrality, this algorithm behaves like a FIHC.

3.2 Guiding the Search over Neutral Networks

Instead of randomly choosing the next solution to explore, the selection can be guided. Indeed, on a NN, $|S| > 1$ solutions can be explored. Only one has to be chosen in order to evaluate one of its neighbors. To do so, we here propose to use the evolvability of solutions.

Algorithm 1 presents the general framework of VEGAS. All the evaluated solutions of the current NN are recorded in S. Then, a *select* method returns a solution $s \in S$. A new neighbor $s' \in \mathcal{N}(s)$ is evaluated (without repetition). If s' has a better fitness value, the set S becomes the singleton $\{s'\}$. Otherwise, if s' has the same fitness value than the current fitness value, it is added to S. A reward is computed for each solution from S, and the *select* method is applied.

The *select* method is one of the main component of VEGAS. For instance, if *select(S)* always returns the last neutral solution evaluated, this algorithm behaves like a NC. Here, we consider that the solution with the highest score, as given by equation (1), is selected. Thus, every time a new solution s' is evaluated in the neighborhood of a solution $s \in S$, s' is recorded to update the score values of all solutions in S according to the credit assignment under consideration. The reward is based on the AUC (see Section 2.3). The arms are the solutions from S, and the fitness values of the evaluated neighbors are used to compare solutions. The AUC gives a way to compare the evolvability of evaluated solutions on a NN. For instance, when the fitness value of neighbors from solution $s \in S$ are better than those of solution $s' \in S$, the AUC of s is higher than the one of s'.

The parameter C in (1) allows to tune the trade-off between exploration and exploitation. Here, it affects the exploration and the exploitation of the neighborhood of solutions from the NN. When C is large, it gives more weight to exploration: the search promotes the sampling of neighborhoods with few solutions evaluated. When C is small, it gives more weight to exploitation: the search promotes the sampling of neighborhoods where the best neighbors have been evaluated so far. This is based on the assumption that solutions with a higher evolvability are more likely to find a portal.

4. EXPERIMENTS

4.1 NK-Landscapes with Neutrality

The family of NK-landscapes is a problem-independent model used for constructing multimodal landscapes [11]. Such a model is of high interest in order to design new search approaches. Parameter N refers to the number of bits in the search space (*i.e.* the binary string length), and K to the number of bits that influences a particular bit from the string (the epistatic interactions). By increasing the value of K from 0 to $(N-1)$, NK-landscapes can be gradually tuned from smooth to rugged. The fitness function (to be maximized) of a NK-landscape $f_{NK} : \{0,1\}^N \to [0,1)$ is defined on binary strings of size N. An 'atom' with a fixed epistasis level is represented by a fitness component $f_i : \{0,1\}^{K+1} \to [0,1)$, associated with each bit $i \in N$. Its value depends on the allele at bit i and also on the alleles at K other epistatic positions (K must be defined between 0 and $N-1$). The fitness $f_{NK}(x)$ of a solution $x \in \{0,1\}^N$ corresponds to the mean value of its N fitness components f_i: $f_{NK}(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x_i, x_{i_1}, \ldots, x_{i_K})$, where $\{i_1, \ldots, i_K\} \subset \{1, \ldots, i-1, i+1, \ldots, N\}$. In the original NK-landscapes, the fitness components are uniformly distributed in the range $[0,1)$, so that it is very unlikely that the same fitness value is assigned to two different solutions. In other words, the neutrality is *null*.

In our study, we will use an extension of this initial model in which neutrality has been added. The way the neutrality is artificially included has an important impact on the structure of the resulting landscapes. Several models of neutrality have been proposed to generalize the initial NK-landscapes by adding a tunable level of neutrality. Among others, there are the NK_p-landscapes (p is for probabilistic) [3], and the NK_q-landscapes (q is for quantized) [15]. NK_p-landscapes are very similar to NK-landscapes, except that the fitness contribution are null with the rate p.In the NK_q-landscapes, that will be used in the paper, the fitness contributions are integer values belonging to the range $[0,q)$. The total fitness is scaled by a factor $1/(q-1)$ in order to translate it in the range $[0,1]$. As indicated by Geard et al. [9] in their comparison of neutral landscapes, NK_q-landscapes are similar to the NK-landscapes in several aspects. The NK_q-landscapes look like NK-landscapes in which rugged hillsides have been flattened into plateaus. The smaller q, the higher the level of neutrality.

4.2 Experimental Design

In order to study the performance of the proposed VEGAS algorithm, we compare it with three other approaches:

- FIHC: a First-Improvement Hill Climbing algorithm, that *strictly* improves the current fitness value during the search;
- NC: a Netcrawler algorithm [4], that allows neutral moves to be performed during the search.
- F2NS: a Fair Neutral Network Search algorithm, that evaluates a random neighbor (without replacement) of a random solution from the NN.

We experiment these algorithms on randomly-generated NK_q-landscapes with $N = 64$, $K \in \{2,4,6,8\}$ and $q \in \{2,3,4\}$. They will give us the opportunity to compare these algorithms according to different configurations with respect to neutrality and non-linearity. The neighborhood is defined with the bit-flip operator of one bit. For every instance, 100 independent executions are performed.

All the algorithms start with a random solution. The stopping condition is given in terms of a maximal number of evaluations, set to 10^5. The four algorithms could converge before the stopping condition. Thus, they have all been included in a random-restart framework: when an algorithm seems to have converged, the search restarts with a new random solution (keeping the best ever found solution). For the FIHC, the search stops when the current solution is a local optimum (no neighbor has a strictly higher fitness). For the NC, the search stops on solutions which are strict local optima where the current can not be strictly improve. For the NC, we fix a second maximal number of evaluations denoted by k. This number has been set according to K and q from preliminary experiments. For every instance, 30 independent runs have been performed. For each run, the number of evaluations needed to converge is recorded. The maximum of this number over the 30 is the value of k (given in Table 1). For the F2NS and VEGAS, we consider that the search converged when S cover the whole NN, and no portal is available on this NN (local optimum plateau). The VEGAS algorithm has a single problem-independent parameter: C, which allows to control the trade-off between the exploration and the exploitation of solutions neighborhood from the NN. Following [7], multiple C-values are investigated: $C \in \{0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 500\}$. Let VEGAS$_n$ denote a VEGAS instance with $C = n$.

Table 1: Maximum number of moves k on the same NN for NC.

q \ K	2	4	6	8
2	23,772	27,950	7,733	6,143
3	1,891	1,648	1,987	1,921
4	8,198	2,000	3593	1189

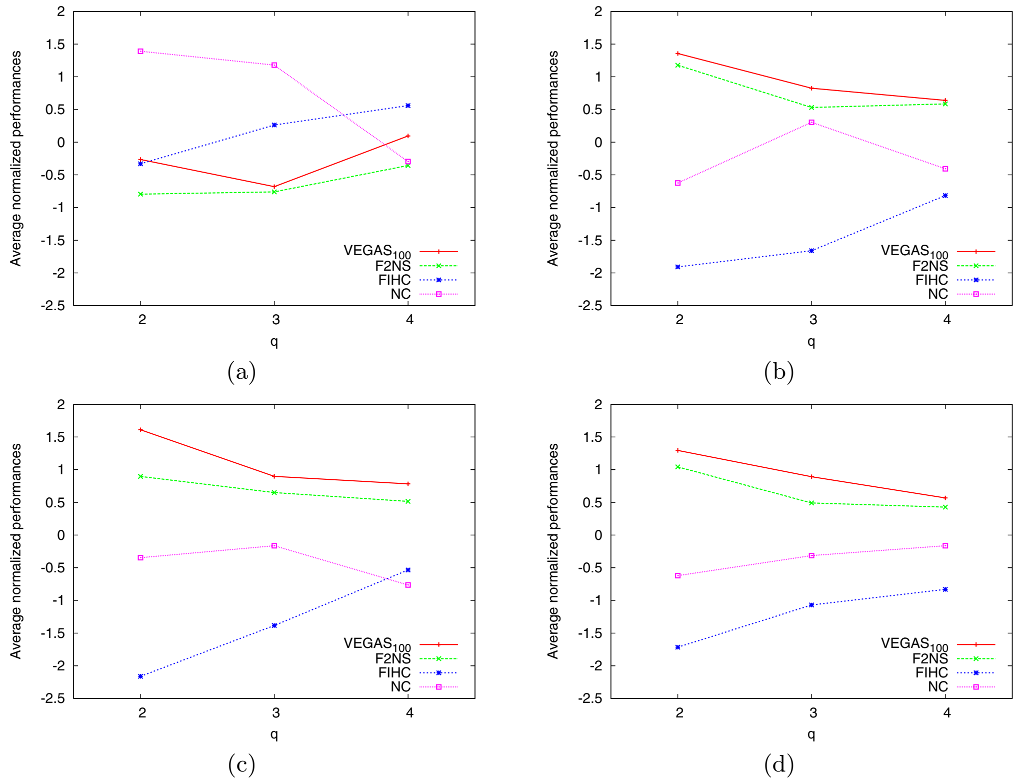


Figure 2: Average normalized fitness found according to parameter q after 10^5 evaluations for (a) $K = 2$, (b) $K = 4$, (c) $K = 6$, (d) $K = 8$. Results for VEGAS$_{100}$, F2NS, FIHC and NC.

4.3 Experimental Results and Discussion

The following experiments first deal with the overall performance of the VEGAS algorithm against F2NS, FIHC and NC. Next, the influence of the parameter C is deeply analyzed and first conclusions on the link between C-values and the dynamics of the algorithm are given.

4.3.1 Performance Analysis

In this section, the four algorithms under study are compared with each other. As far as VEGAS is concerned, only one parameter (C) is to be set. The influence of this parameter is studied in the next section. Here we use a value of $C = 100$, as it leads to overall good performance.

With respect to the non-linearity (K) and to the level of neutrality (q), the fitness values of the solutions found are not comparable. Hence, in order to compare the performance of the four algorithms according to the parameters K and q, the fitness values have been normalized using the average and the standard deviation of all fitness values found for the same problem instance. Such an approach brings the average performance around zero and enlarges the extreme behaviors. The average value $\bar{f}$ and the standard deviation σ_f of all fitness values are computed with the 100 runs performed by each algorithm. For a fitness value f, its normalized value $\tilde{f}$ is set to: $\frac{f-\bar{f}}{\sigma_f}$. The performance of a given algorithm on a particular NK_q-landscape is given in terms of this average $\tilde{f}$-value over the 100 executions.

The respective performance of all the algorithms is given in Figure 2. For each K-value, the performance is plotted with respect to q. For $K \in \{4, 6, 8\}$, VEGAS$_{100}$ and F2NS clearly outperform FIHC and NC. The comparison of the performance of VEGAS and F2NS is more difficult as, even if VEGAS always outperforms F2NS, both are very close and the difference may not be statistically significant (see below). For $K = 2$, Figure 2 (a) does not show a clear result. Indeed, for $q = \{2, 3\}$, NC gives the best performance, but it becomes almost the worst approach for $q = 4$, where the best approach seems to be FIHC.

In order to assess these results, a Wilcoxon two-sample paired signed rank test is used with the null hypothesis that the median performance of the paired differences of the two algorithms under comparison is null. Table 2 gives the output of the Wilcoxon test.

These results confirm that:

- Except for $K = 2$, VEGAS and F2NS always significantly outperform FIHC and NC. It shows that (i) it is worth exploring the NN (in contrast to FIHC), and that (ii) the way to do it has a large influence. Indeed, even if a method selects a solution at random on the NN (like F2NS), it may outperform a method that always selects the last-evaluated solution (like NC).

- VEGAS is never outperformed by F2NS. It shows that guiding the search over a NN allows to obtain better solutions.

In summary, exploring NN is a good way to guide the search over neutral landscapes, at last for a reasonable level of neutrality. However, this must be done carefully, and it

Table 2: Wilcoxon paired tests on the 100 runs between the 4 algorithms. $=$ means both algorithms are not significantly different, $>$ means the algorithm of the row outperforms the one of the column and $<$ means for the contrary.

		q=2			q=3			q=4		
		VEGAS	F2NS	FIHC	VEGAS	F2NS	FIHC	VEGAS	F2NS	FIHC
$K = 2$	F2NS	$=$			$=$			$<$		
	FIHC	$=$	$=$		$<$	$<$		$>$	$>$	
	NC	$>$	$>$	$>$	$>$	$>$	$>$	$<$	$=$	$<$
$K = 4$	F2NS	$=$			$<$			$=$		
	FIHC	$<$	$<$		$<$	$<$	$-$	$<$	$<$	
	NC	$<$	$<$	$>$	$<$	$<$	$>$	$<$	$<$	$>$
$K = 6$	F2NS	$<$			$=$			$=$		
	FIHC	$<$	$<$		$<$	$<$		$<$	$<$	
	NC	$<$	$<$	$>$	$<$	$<$	$>$	$<$	$<$	$=$
$K = 8$	F2NS	$=$			$<$			$=$		
	FIHC	$<$	$<$		$<$	$<$		$<$	$<$	
	NC	$<$	$<$	$>$	$<$	$<$	$>$	$<$	$<$	$>$

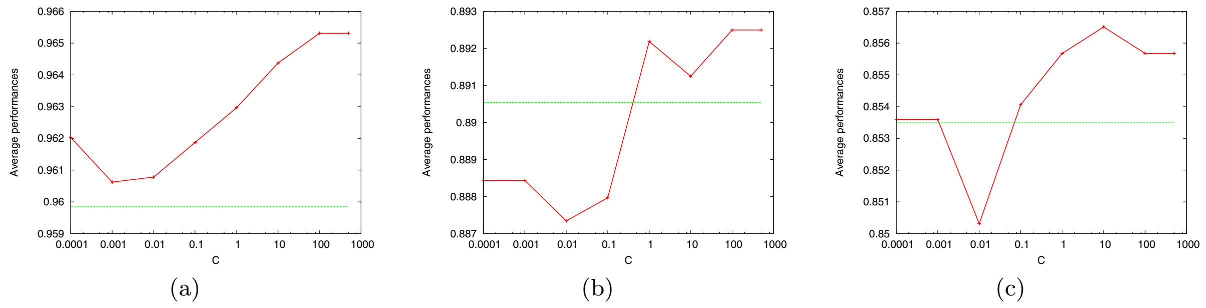


Figure 3: Average fitness found by VEGAS$_C$ as a function of C. $K = 6$ **(a)** $q = 2$ **(b)** $q = 3$ **(c)** $q = 4$**. The horizontal dotted line is the performance of F2NS.**

seems to be better to randomly choose the next solution to explore than making always the same arbitrary choice. An alternative that shows interesting results is to pursue the search from the solution with the best evolvability.

4.3.2 *Impact of Parameter C*

In this section, we analyze the performance of VEGAS according to the setting of its (single) problem-independent parameter: C. Comparing the results obtained for all the instances, the efficiency of VEGAS does not seem to be clearly sensitive to the parameter C. This is confirmed by the Wilcoxon paired test that indicates that no general trend can be found. However, for some instances, VEGAS with exploration ($C > 1$) outperforms VEGAS with exploitation ($C < 1$) in average.

This is illustrated in Figure 3 for $K = 6$, that gives the average fitness values obtained according to parameter C. Similar figures were obtained for $K = 4$ and $K = 8$. Indeed, the exploration of NN ($C > 1$) gives better results than their exploitation ($C < 1$). In Figure 3, there is also a dotted line representing the average fitness value obtained with F2NS (when a solution is chosen at random among the NN). This confirms that VEGAS$_{100}$ outperforms F2NS, but also that both methods obtain good performance, F2NS may produce better results than some versions of VEGAS, typically when $C < 1$.

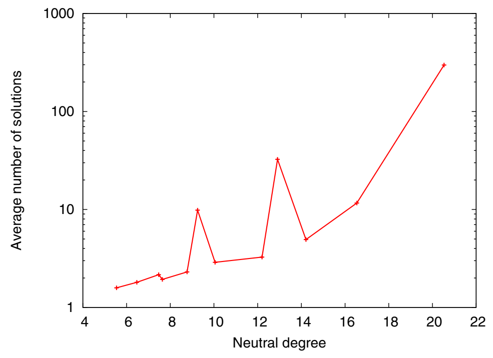


Figure 4: Average number of evaluated solutions per NN with respect to the neutral degree.

.

4.3.3 *Impact of Neutrality*

Table 3 gives the average value, over 10 000 random solutions, of the neutral degree according to the NK_q-landscape under study. The neutrality decreases when K and/or q increase. Figure 4 gives the average number of solutions evaluated per NN by VEGAS$_{100}$ according to the neutral degree. It shows the influence of the neutrality on the number of

Table 3: Average neutral degree of the NKq instances.

q \ K	2	4	6	8
2	20.53	16.54	14.21	12.2
3	12.91	10.05	8.77	7.64
4	9.25	7.47	6.47	5.54

evaluated solutions on each encountered NN. This number increases exponentially with the neutral degree. Let us indicate that the two picks correspond to $K = 2, q = \{3, 4\}$. As we already pointed out, when $K = 2$ (small epistasis), VEGAS has a different behavior.

4.3.4 Exploration *vs.* Exploitation

Previous experiments show that the exploration of a larger number of solutions from the NN gives, in general, better performance. In order to analyze this in more details, some statistics have been computed to study the dynamics of the search. Two main statistics are computed here: (*i*) the number of NN evaluated, (*ii*) the number of solutions evaluated on each NN, which corresponds to the size of the neutral networks explored part.

Figure 5 gives (a) the average number of NN and (c) the average number of solutions evaluated on each NN according to C. First, there is a clear difference between VEGAS with exploitation and VEGAS with exploration. Indeed, for $C \geq 1$, average values are similar. The same happens for $C < 1$. The different curves may be cut around $C = 0.5$ into two homogeneous parts. Second, the higher the average number of NN, the smaller the average number of evaluated solutions *per* NN. This attests a large difference on the behavior of the VEGAS algorithm regarding the C-values. In other words, when C is turned to exploration, more NN are explored, but the portion of evaluated solutions is small. When C is turned to exploitation, few NN are explored, but they are deeply evaluated. Such dynamics may explain that VEGAS with exploration gives, in some cases, a better performance than VEGAS with exploitation.

Figure 5 (b) and (d) also gives the same values with respect to q for different algorithms: VEGAS with exploration (VEGAS$_{100}$), VEGAS with exploitation (VEGAS$_{0.01}$) and F2NS (random choice). Let us remark that similar trends happen for $K \in \{4, 8\}$. It appears that the F2NS approach has a behavior "in-between" the two VEGAS algorithms. As we previously seen on Figure 3, the performance of F2NS is either below VEGAS or "in-between" the two VEGAS variants. A natural conclusion is that this balance between exploration and exploitation is a critical issue for the performance of the algorithm.

5. CONCLUSIONS AND FUTURE WORKS

This work proposes a new methodology to deal with neutral combinatorial optimization problems. In this approach, all solutions identified on a plateau are considered in order to help the search to progress. Then, the most promising solution evaluated on the plateau is selected adaptively, based on the evolvability of solutions. VEGAS is an adaptive search algorithm using the multi-armed bandit framework and the 'area under the curve' credit assignment principle [7].

An experimental study on NK-landscapes with neutrality has been conducted. It first shows that randomly choosing a solution on the plateau outperforms a netcrawler-based multi-start local search for a reasonable level of neutrality. The experimental analysis also shows that VEGAS generally gives better results than selecting a solution at random on the plateau. The VEGAS dynamics is different depending on the level of neutrality. Moreover, VEGAS requires a single problem-independent parameter, that allows to tune the trade-off between the exploration and the exploitation of the plateau. The influence of this parameter on the search dynamics has been deeply analyzed.

This approach shows encouraging results and open future research directions. As in adaptive operator selection [7], we need to test others credit assignment methods, probably more specific to neutral landscapes. Moreover, similar experiments will allow to better understand the dynamics of the VEGAS algorithm on other combinatorial optimization problems where neutrality arises, such as in flowshop scheduling.

6. REFERENCES

[1] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47:235–256, May 2002.

[2] W. Banzhaf. Genotype-phenotype-mapping and neutral variation - a case study in genetic programming. In *PPSN III: Third Conference on Parallel Problem Solving from Nature*, p322–332, London, UK, 1994. Springer-Verlag.

[3] L. Barnett. Ruggedness and neutrality - the NKpfamily of fitness landscapes. In C. Adami, R. K. Belew, H. Kitano, and C. Taylor, editors, *ALIFE VI, Proceedings of the Sixth International Conference on Artificial Life*, p18–27. ALIFE, The MIT Press, 1998.

[4] L. Barnett. Netcrawling, optimal evolutionary search with neutral networks. In *Proceedings of the 2001 Congress on Evolutionary Computation, CEC 2001*, p30–37. IEEE Press, 2001.

[5] L. Da Costa, A. Fialho, M. Schoenauer, and M. Sebag. Adaptive operator selection with dynamic multi-armed bandits. In M. K. et al., editor, *GECCO'08: Proc. 10th Annual Conference on Genetic and Evolutionary Computation*, p913–920. ACM Press, July 2008.

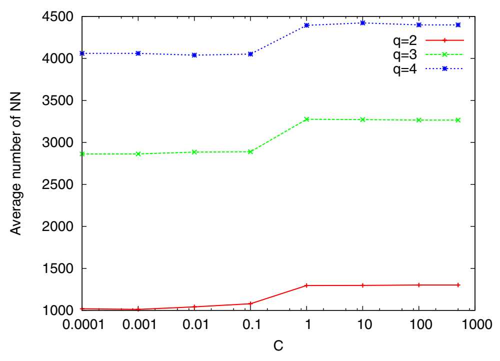
[6] A. E. Eiben, Z. Michalewicz, M. Schoenauer, and J. E. Smith. Parameter control in evolutionary algorithms. In *Parameter Setting in Evolutionary Algorithms*, p19–46. 2007.

[7] A. Fialho, M. Schoenauer, and M. Sebag. Toward comparison-based adaptive operator selection. In J. B. et al., editor, *GECCO'10: Proc. 12th Annual Conference on Genetic and Evolutionary Computation*, pages 767–774. ACM Press, July 2010.

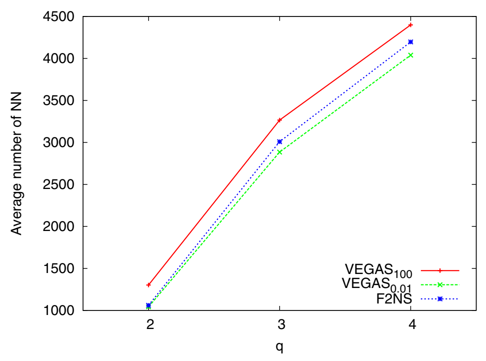
[8] J. Frank, P. Cheeseman, and J. Stutz. When gravity fails: local search topology. *Journal of Artificial Intelligence Research*, 7:249–281, 1997.

[9] N. Geard, J. Wiles, J. Hallinan, B. Tonkes, and B. Skellett. A comparison of neutral landscapes - nk, nkp and nkq. In *Proceedings of the Congress on Evolutionary Computation, 2002. CEC '02.*, p205–210, 2002.

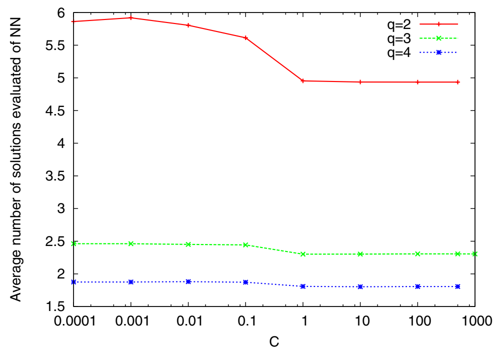
[10] I. Harvey and A. Thompson. Through the labyrinth evolution finds a way: a silicon ridge. In T. Higuchi,



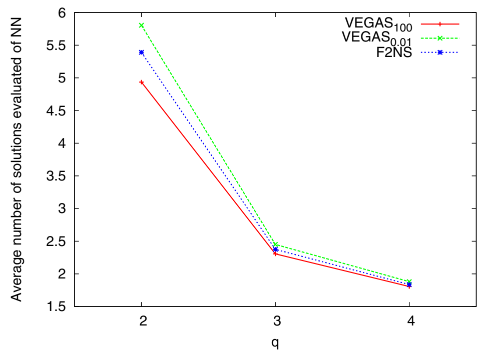
(a)



(b)



(c)



(d)

Figure 5: The average number of NN (top) and the average number of evaluated solutions of each NN (bottom). On the left, the average quantities are compared according to C according to the parameter q. On the right, they are compared according to q between 2 VEGAS algorithms (with $C = \{100, 0.01\}$) and F2NS. for all plots, $K = 6$.

M. Iwata, and W. Liu, editors, *Evolvable Systems: From Biology to Hardware, First International Conference, ICES 96*, volume 1259 of LNCS, p406–422. Springer, Berlin, 1996.

[11] S. A. Kauffman. *The Origins of Order*. Oxford University Press, New York, 1993.

[12] M. Kimura. *The neutral theory of molecular evolution*. Cambridge University Press., 1983.

[13] M.-E. Marmion, C. Dhaenens, L. Jourdan, A. Liefooghe, and S. Verel. NILS: a Neutrality-based Iterated Local Search and its application to flowshop scheduling. In *European Conference on Evolutionary Computation in Combinatorial Optimisation (EvoCop'2011)*, LNCS. Springer-Verlag, 2011.

[14] M.-E. Marmion, C. Dhaenens, L. Jourdan, A. Liefooghe, and S. Verel. On the neutrality of flowshop scheduling fitness landscapes. In *Learning and Intelligent OptimizatioN (LION 5)*, LNCS. Springer-Verlag, 2011.

[15] M. Newman and R. Engelhardt. Effect of neutral selection on the evolution of molecular species. *Proc. R. Soc. London B.*, 256:1333–1338, 1998.

[16] P. Smith, T.and Husbands, P. Layzell, and M. O'Shea. Fitness landscapes and evolvability. *Evol. Comput.*, 10(1):1–34, 2002.

[17] T. M. C. Smith, P. Husbands, and M. O'Shea. Neutral networks in an evolutionary robotics search space. In *Congress on Evolutionary Computation, CEC 2001*, p136–145. IEEE Press, 2001.

[18] P. F. Stadler. *Towards a theory of landscapes*, volume 461, p78–163. Springer Berlin / Heidelberg, 1995.

[19] T. Stewart. Extrema selection: Accelerated evolution on neutral networks. In *Congress on Evolutionary Computation CEC2001*, p25–29. IEEE Press, 2001.

[20] D. Thierens. An adaptive pursuit strategy for allocating operator probabilities. In *Conference on Genetic and evolutionary computation*, GECCO '05, p1539–1546, New York, NY, USA, 2005. ACM.

[21] V. K. Vassilev and J. F. Miller. The advantages of landscape neutrality in digital circuit evolution. In Springer, editor, *3rd International Conference on Evolvable Systems: From Biology to Hardware. LNCS.*, volume 1801, p252–263, 2000.

[22] S. Vérel, P. Collard, M. Tomassini, and L. Vanneschi. Fitness landscape of the cellular automata majority problem: view from the "Olympus". *Theor. Comp. Sci.*, 378:54–77, 2007.

[23] A. Wagner. *Robustness and Evolvability in Living Systems*. Princeton Uiversity Press, 2005.

[24] T. Yu and J. F. Miller. Neutrality and the evolvability of Boolean function landscapes. In *Eurogp01, 4th European Conference on Genetic Programming*, p204–217. Springer, Berlin, 2001.