University "Alexandru Ioan Cuza" of Iaşi
Faculty of Computer Science

TECHNICAL REPORT

# Walking the Royal Road with Integrated-Adaptive Genetic Algorithms

**Ovidiu Gheorghieş**
**Henri Luchian**
**Adriana Gheorghieş**

**TR 05-04,** July 2005

**ISSN 1224-9327**

**Universitatea "Alexandru Ioan Cuza" Iaşi**
**Facultatea de Informatică**
Str. Berthelot 16, 700483 -Iaşi, Romania
Tel. +40-232-201090, email: bibl@infoiasi.ro

# Walking the Royal Road with Integrated-Adaptive Genetic Algorithms

Ovidiu Gheorghieş
Faculty of Computer Science
"Al. I. Cuza" University of Iaşi
Iaşi, Romania
ogh@infoiasi.ro

Henri Luchian
Faculty of Computer Science
"Al. I. Cuza" University of Iaşi
Iaşi, Romania
hluchian@infoiasi.ro

Adriana Gheorghieş
Faculty of Computer Science
"Al. I. Cuza" University of Iaşi
Iaşi, Romania
adrianaa@infoiasi.ro

## ABSTRACT

The aim of this paper is to show that exploiting knowledge extracted from the optimization process is important for the success of an evolutionary solver. In the context of NK fitness landscapes, we identify two causes for the difficulty of an optimization problem: the intrinsic combinatorial difficulty and the random-search hybridization. We apply these concepts for the royal road fitness landscape. Experimental results indicate that Integrated-Adaptive Genetic Algorithms (IAGA) are particularly suited for tackling random-search hybridization. A learn-as-you-go system aimed at a fine-grained adaptation of operators behavior increases the solving power and convergence speed of IAGA. We conclude that the royal road problem is actually being "royal" not for the traditional GA, but for a class of adaptive genetic algorithms.

## 1. INTRODUCTION

Knowledge is a terrible thing to waste. Disregarding much potentially useful information on how or why the improvement of solutions takes place is no longer a viable option for modern genetic algorithms.

There are many areas within a genetic algorithm which may benefit from the exploitation of search-derived knowledge. For example, the number of times an operator is to be applied may not have one optimal value which is appropriate during the entire run of the genetic algorithm. Also, the specific way in which an operator works may also need to be adapted as evolution progresses. The important issue is not however finding the appropriate settings for a particular state during the optimization process, nor for the entire run for a particular problem instance, and neither for all the instances of a particular problem. To avoid having to start over the tedious process of parameter tuning each time a new problem is tackled, one would prefer employing a system which adapts itself to the particularities of the problem at hand.

Whenever an adaptive system is used, it is important to establish its capability of contributing to the success of the optimization process. In this paper we address this issue in the context of NK fitness landscapes. It has been shown that it is easy to construct via a stochastic method an NK fitness landscape in which finding the optimum is an NP-complete problem. It has also been proven that particular cases of NK fitness landscapes ("royal road" included) can be solved polynomially. Despite of this, the royal road problem has a long standing record for being difficult for GAs. In trying to identify the causes, we evince two aspects of problem difficulty: intrinsic combinatorial difficulty and random-search related difficulty.

We show how the problem of finding the optimum in a particular case of NK fitness landscapes can be transformed into the NP-hard problem of finding the maximal stable set in a graph. Using this transformation we show that the royal road problem has a trivial intrinsic combinatorial difficulty and attribute its proven difficulty to its hybridization with random-search.

The royal road problem is used to provide a tunable degree of random search hybridization for an easy to analyze NK fitness landscape. We experiment on it using the general adaptive framework provided by Integrated-Adaptive Genetic Algorithms (IAGA) [17]. We add to IAGA a learn-as-you-go system which allows operators to further adapt themselves by inspecting the immediate results of their applications (IAGA$^+$). IAGA improves on the traditional GA, while IAGA$^+$ improves on IAGA in terms of how fast a solution is found. Also, the more relevant adaptation we use, the harder are the problem instances which can be actually solved. Experimental results compared on problems with the same intrinsic combinatorial complexity indicate that the adaptive systems employed are useful in tackling the complexity caused by random-search hybridization.

### 1.1 Related Work

Adaptive parameter control methods are classified as deterministic, adaptive and self-adaptive [5]. Deterministic systems adjust some GA parameter values according to a fixed, predefined law [2]. Adaptive control uses feedback from the search process in order to find out how the parameter values are to be changed [14], [23]. Self-adaptive control attaches

to chromosomes additional information which provides some guidance to the operators [3].

According to [18], most efforts on parameter control are focused on adapting only a few elements of the genetic algorithm at a time. Systems which adapt a wider range a parameters within a GA are however exhibiting some attractive properties: [13], [22].

NK fitness landscapes were introduced by Kaufmann [15] and have been since extensively studied [1]. The decision problem associated to this fitness landscape is NP-complete, thus its study can provide valuable insight into the behavior of any optimization procedure which tackles such type of problem [10].

## 1.2 Paper Outline

In section 2 we present the NK fitness landscapes and the royal road function as a particular case; we informally show how the problem of finding the optimum in an NK fitness landscape can be translated into the problem of finding the maximal stable set in a graph. Using this transformation we dissociate the difficulty of a problem into its intrinsic combinatorial complexity and the one caused by random-search hybridization. Section 3 presents the architecture of integrated-adaptive genetic algorithms (IAGA). We propose an extension to this framework called IAGA$^+$, which uses a "learn-as-you-go" module aimed at atomically improving operators (subsection 4.3). In section 4 the traditional GA, IAGA and IAGA$^+$ are run against royal road problems with identical combinatorial difficulty, but an ever increasing degree of random-search hybridization. Section 5 conjectures that adaptive systems are particularly suited for managing random-search related problem difficulty and gives hints on further study. To facilitate replication and application to other real-world problems, the source code we used for this paper will be made freely available on the authors' web page.

# 2. BACKGROUND

NK fitness landscapes are fitness functions on bitstring of $N$ genes, $(X_1 \dots X_N)$, parametrized to allow interactions between $K$-uples of genes so as to make them tunably "rugged". One way of formulating such fitness function is given below:

$$F(X) = \sum_{i=1}^{f} \frac{1}{f} F_i(X_{j_1^i}, X_{j_2^i}, \dots, X_{j_K^i}) \qquad (1)$$

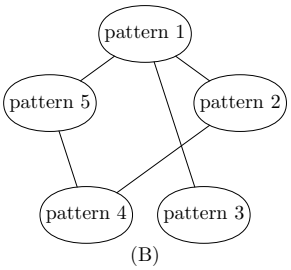
where $f$ represents the number of fitness components and $K$ the (fixed) number of genes affecting the fitness component $F_i$ ($i \in \{1 \dots f\}$). The genes which affect a given fitness component are said to form a neighborhood.

It has been proven by Weinberger that the NK optimization problem with random neighborhood is $\mathcal{NP}$-complete for $K \geq 3$. However, when adjacent neighborhoods are considered, the problem is solvable in $\mathcal{O}(2^K N)$ steps, and thus is in $\mathcal{P}$ [24].

Let us consider a particular form for fitness components $F_i$, which gives 1 if $X_{j_1^i}, X_{j_2^i}, \dots, X_{j_K^i}$ fit a given pattern $p_i$ and 0 otherwise. Such a pattern (or schema) $p_i$ has fixed values on positions $j_1^i, j_2^i, \dots, j_{m_i}^i$ and the "don't care" symbol "$*$" elsewhere. In the remainder of the paper we will discuss in the context of such pattern-based fitness components. Figure 1 (A) contains a series of patterns which can be used to define such an NK fitness landscape. Some patterns agree on the same value for all their defined positions, while others are in conflict. In order to maximize the resulting fitness function $F$ one must find an assignment for the gene values which maximizes the number of fit patterns.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| pattern 1: | 1 | 0 | 0 | * | * | * | * | * |
| pattern 2: | * | * | 1 | 0 | 1 | * | * | * |
| pattern 3: | 0 | * | * | * | 1 | 1 | * | * |
| pattern 4: | * | * | * | 1 | * | * | 1 | 1 |
| pattern 5: | * | * | 1 | * | * | 1 | * | 0 |
| chromosome: | $X_1$ | $X_2$ | $X_3$ | $X_4$ | $X_5$ | $X_6$ | $X_7$ | $X_8$ |

(A)




(B)

**Figure 1: (A) Patterns defining an NK fitness landscape. (B) Associated pattern conflict graph ("essence graph").**

In figure 1 (B) the patterns are modeled as graph nodes and pattern conflicts are represented graph edges. Thus, the problem of finding the optimum in this type of NK fitness landscape can be reformulated as to finding the maximal stable set in a graph. We call this problem the "essence problem" of an NK fitness landscape.

Although the maximum stable set problem is NP-hard [7], there are classes of graphs for which this problem can be solved in polynomial time. Examples of such classes are perfect graphs [11] and $K_{1,3}$-free graphs [19]. Such properties induce what we call "intrinsic combinatorial difficulty" of the associated NK fitness landscape problem.

Let us consider adjacent, non-overlapping neighborhoods for the genes contributing to each fitness component $F_i$. If every pattern $p_i$ has only 1-s on the defined positions, we obtain a formulation of the "royal road" problem [20]. For this problem, the associated pattern conflict graph has no edges. To solve this "essence problem", one must find the maximal stable set in a graph with no edges, which is trivial! In the next subsection we sketch an explanation for the proven difficulties encountered by optimization methods when tackling this type of problem.

## 2.1 Intrinsic Difficulty and Random-Search Hybridization

| pattern 1 | | | pattern 2 | | | pattern 3 | | |
|---|---|---|---|---|---|---|---|---|
| 1 | * | * | * | 1 | * | * | * | 1 |
| 11 | ** | ** | ** | 11 | ** | ** | ** | 11 |
| 111 | *** | *** | *** | 111 | *** | *** | *** | 111 |

**Figure 2: Example of random-search hybridization for the patterns (pattern 1, pattern 2, pattern 3). Resulting NK fitness landscape is increasingly harder, but the essence problem remains the same.**

We have shown how finding the optimum in an NK fitness landscape can be translated into finding the maximal stable set in the "essence graph". This graph encapsulates the *intrinsic combinatorial difficulty* of the problem of finding the optimum in the NK fitness landscape. Any algorithm which finds the solution in an NK fitness landscape cannot be more efficient than the best algorithm which finds the maximal stable set in the essence graph.

However, the problem of finding the optimum in an NK fitness landscape can be made significantly harder from a practical viewpoint, while its essence problem remains unchanged (and thus does not increase in difficulty). For the sake of simplicity we show how this can be done for the royal road problem, but a similar transformation can be performed on any pattern-based NK fitness landscape (which may additionally exhibit some pattern conflicts). Let us consider the following patterns: (1**, *1*, **1) which, as shown above, create a NK fitness landscape whose essence graph has no edges. For any individual pattern we can expand the values on fixed positions into an arbitrary larger sequence of fixed positions, such that the pattern-conflict relationships are preserved. An example of this process is illustrated in figure 2: we call it *random-search hybridization* of a problem.

Random-search hybridization can be used to increase the difficulty of a (potentially easy) problem by simply enlarging the search space. Subsection 2.2 gives more insight into the dependency between the population size and the dimension of the search space.

## 2.2 Random-search hybridization and population size

Theoretical results to guide population sizing in GAs are few and somewhat difficult to apply in practice. Two methods have been described by Goldberg. In the first one [8] the population is sized for optimal schema processing, and in the the second one [9], optimization was performed for accurate schema sampling. Goldberg's work indicate that the optimal size for binary-coded strings grows exponentially with the length of the string. Reeves [21] tried to identify lower bounds for population size in GAs. The principle adopted by him was that every possible point in the search space should be reachable from the initial population by crossover only. Reeves concluded that the minimal population size depends on the alphabet cardinality and the string length. For larger alphabets, the minimal population sizes are substantial even for short strings.

As the degree of random-search hybridization increases, it becomes intractable to create a population so large that all patterns can be synthesized via crossover. Thus, using some form of mutation(s) becomes a necessity. All operators (whether unary, binary or multi-ary) may also have destructive effects. In this context, the event of synthesizing a pattern is not to be treated lightly ("find and forget"): it is important to capitalize on the occasional "luck" and learn to do quickly something that could potentially take very long to happen again.

More importantly, one must take into account the importance of negative experience. Whenever an operator destroys a pattern, it is relevant to learn that (rather than to "destroy and forget"). It can be argued that copies of the pattern can be found within other individuals, but this may well not happen in a bounded population exploring a large search space. After all, everybody wants to avoid making the same mistakes again and again.

We argue that to "find and forget" and to "destroy and forget" can cause GA's poor performance, at least when random-search hybridization is involved. Our experiments show that these issues can be competently [16] addressed by an evolutionary system which uses broad adaptive mechanisms.

# 3. INTEGRATED-ADAPTIVE GENETIC ALGORITHMS

Integrated-adaptive genetic algorithms (IAGA) define a general schema of a genetic algorithm which is able to dynamically adapt both the rate and the behavior for each of its operators. The rates of the operators are adapted based on their recorded merits (the better an operator proves itself to be, the more often will it get applied in subsequent generations). Also, the specific way in which operators work co-evolves with the main GA population, such that presumably better operator variants emerge.

It is instructive to present a comparison between IAGA and parameter-less GAs [12]. These systems are both motivated by the observation that from the user's point of view the setting of parameters is a far from a trivial task. However similarities end here.

Parameter-less GAs prescribes the crossover rate and (at least in the initial version) ignores mutation altogether. IAGA explicitly acknowledges that in a real world GA there may be more types of crossover operators and mutations operators. The rates of these operators are not fixed, but adjusted according to their proven performance. The crossover in the parameter-less GA creates offspring according to a predefined, fixed algorithm. In IAGA all operators are automatically and flexibly customized in such way that they presumably become suited for the particular state the optimization process is in.

The techniques in parameter-less GAs can be applied only for problems where specific mathematical properties can be inferred. IAGA defines a framework which is compatible with virtually any traditional GA that can be designed for a given problem, only adding an extra-layer of adaptiveness. Everything else being equal, this layer adds a negligible constant-time overhead per generation in order to perform

due adjustments to parameter values.

The population size is acknowledged by many as a critical GA parameter [6]. The original definition of IAGA follows the line of the traditional GAs in keeping the population size constant, while the evolution of this parameter is the strong point of parameter-less GAs. It would be interesting to also add to IAGA a parameter size control component; in this paper however we will be focusing on adapting operator-related parameters as a tractable countermeasure to random-search hybridization.

A real world GA may use several types of operators, whether unary (mutations), binary (crossovers), or multi-ary [4]. Every such operator consists in an algorithm (called *operator type*) whose exact behavior is controlled by a specific number of parameters. Actual genetic operator are obtained by assigning specific values to the parameters. These operator are then used by the GA to create new offspring.

It is typical for a GA designer not to know beforehand which parameter values will better suit the optimization process. To explore the possible settings, IAGA uses a (small) population of actual operator for each operator type. By letting these actual operators compete for breeding better candidate solutions, the optimization procedure obtains feedback on the on the effectiveness of various parameter settings.

Each actual operator has a specific rate which ultimately results in the number of times the operator is applied in one generation. Again, from a practical standpoint, it is virtually impossible to set them *a priori* to effective values. IAGA is based on the idea that the operator rates should be updated according to the efficiency the operators demonstrate during the optimization process. This adaptive system is called *rate adaptive system* (RAS).

Algorithm 1 gives a schematic overview of IAGA.

**Algorithm 1** Integrated-Adaptive Genetic Algorithm (overview)

```
initialize operators (parameters)
initialize main GA's evolution (operator rates)
while not halting condition do
    run the main GA for one generation
    let Operator Adaptive System (OAS) adapt operators
    let Rates Adaptive System (RAS) adapt operator rates
```

## 3.1 Rates Adaptive System — RAS

The rate adaptive system (RAS) aims at increasing the rates of recently successful operators.

To each actual operator *op* involved in the optimization process an information $\rho(op)$ called *reward* is attached. Its purpose is to record a glimpse of the experience acquired so far during the evolution process regarding the effectiveness of that operator. Recently acquired experience is to be treated as more relevant than the old one, the importance of which fades away gradually.

### 3.1.1 Performance Record

In order to have an estimate on how well an operator performs during the most recent generation, a measure called *performance record* is used. This measure is obtained during the evolution process of the main GA and it is then used by the RAS to adjust the rates of the operators. We present below a domain-independent definition which can be used with no modifications to various concrete applications.

For each application of an actual operator, the following types of events are considered. *Absolute improvement*, when an offspring has a better fitness than the best fitness from the previous generation. *Improvement*, when an offspring has a better fitness than all of its parents, but not one which makes it an absolute improvement. *Plateau walk*, when no offspring is either better than the best of the parents, nor worse than the worse parent. *Worsening*, when some offspring has a worse fitness than the worse parent and no offspring is better than all parents.

The *performance record* of an operator is an uple comprised of the number of each of the events above generated by that operator during one generation. In order to perform rate adjustments, the operators are ordered according to 4-step strategy described below. Since some operators may be applied more times than others during one generation, one has to take into account relative frequencies of a specific event type, and not absolute frequencies.

Operators which yield relatively more absolute improvements are considered to be better. A tie with respect to the absolute improvement rate is broken via the improvement rate: again, more is better. If there is still a draw, the operator that worsens relatively fewer candidate solutions is to be preferred. The fourth rule is that operators which potentially waste computational time producing relatively more plateau walks are to be discouraged.

### 3.1.2 Rewards-Based Rate Adaptation

The reward-based rate adaptation is designed to increase the rate of operators with better performance records.

Suppose that the main GA is about to start generation $t$. By applying the operators in use with their corresponding rates, a new population is obtained. After generation $t$ is completed, each actual operator *op* has its performance record updated. Let $\chi(op)$ denote the number of operators what have a worse performance record than *op*.

Rewards are updated after each main GA generation according to the formula (notations are explained below):

$$\rho(op) \leftarrow \delta\rho(op) + \beta + \gamma\chi(op) \tag{2}$$

Here, $\delta$ is the *diminish* rate, measuring how strong the influence of past experience will be. The *gain* constant $\gamma$ is used for rewarding operators that are better than others (the more operators one operator "defeats", the more it will be reinforced). The *base* constant $\beta$ is used to make sure that no operator will vanish altogether, thus having no more chances of proving its worthiness. Note that $\delta$ and $\gamma$ balance the respective influences of older experience versus most re-

cent experience. Since we consider $\delta \in (0, 1)$, the influence of past experience diminishes as it grows older. For our experiments we have reached a reasonable balance with the settings: $\delta = 0.8$, $\beta = 0.1$, $\gamma = 1.0$.

The operator rates to be used for the next generation of the main GA are then set proportionally to the new reward values.

## 3.2 Operator Adaptive System — OAS

The operator adaptive system aims at creating ever better instantiations of operator types by tuning their parameter values. This process is guided by the experience gained during evolution.

For each operator type, a (typically small) population of assignments for its parameter values is defined. These assignments create actual operators which are directly used by the main GA.

A domain independent strategy for adapting the parameter values is to use a *operator genetic algorithm* (OGA) for each operator type. The population of an OGA consists of chromosomes which represent assignments (parameter values for that operator type). The operators which act on such chromosomes can be defined in a straightforward manner; we provide the source code to clarify this.

OGA's evaluation for a chromosome (parameter values for an operator type) is the performance record of the instantiated operator during the previous generation of the main GA. Technically, it represents a side-effect of the run of the main GA. The result is that the main GA and OGA-s are interlaced, one generation in either of them providing data for the next generation in the other. In effect, the OGA-s and the main GA populations co-evolve in an integrated system.

## 3.3 Integration of RAS and OAS-s

Figure 3 shows how main GA, RAS and OAS can be integrated. Whenever a new operator variant is synthesized, it inherits the reward from its parents (eventually as an average).

RAS and OGAs do not add significantly to the complexity of the main GA: it is easy to design them to work in constant time. Note that the evaluation step in OGAs consists of the mere reading of the performance records given by the main GA.

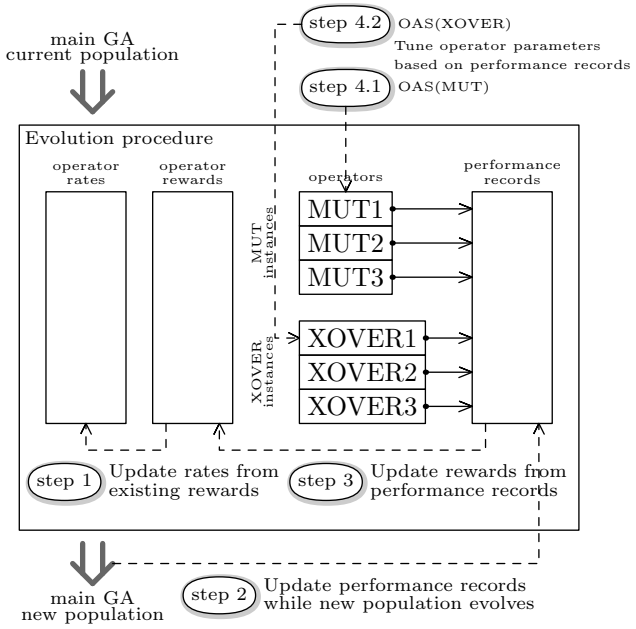



**Figure 3: Integration of the main GA, RAS and OAS. This example assumes the existence of only two types of operators *MUT* and *XOVER*. For each type of operator, we depict three actual operators which are used by the main GA to create offspring.**

# 4. EXPERIMENTAL RESULTS

## 4.1 Overview

The goal of the experiments is to establish the role of adaptive systems in tackling an ever increasing degree of random-search hybridization.

The setup is similar to that of [17], but there are some significant differences. We do not consider the royal road patterns to consist exclusively of 1-s, because this adds an unjustified distortion of the results in favor of the adaptive system. Indeed, it is straightforward for a mutation operator to learn that globally setting 0-s into 1-s is a good thing. The issue is not however that this is easy to learn, but instead easy to guess.

In order to make the results relevant for our purpose, in this paper we apply IAGA to a modified royal road problem; the patterns of the target chromosome are not entirely made of 1-s, but contains some 0-s. Since it is no longer possible to learn globally, per chromosome, which is the preferred way of changing the value of any gene, we will use a mutation operator which is configurable per-locus. Thus, an instance of this operator type tries to evolve optimal probabilities of mutating genes for each locus of the chromosome. This obviously larger search space for parameter values poses new problems which we evince during the experiments.

We propose a system called *learn-as-you-go* which addresses the need for further exploitation of the search-derived information. Thus, the OAS does not rely solely on performance records, but also on the information which can be extracted from each individual application of an operator. We call this extended framework IAGA$^+$.

We present comparative results of the traditional GA (TGA), IAGA and IAGA$^+$ for problems royal-road problems which have 4, 8 and 16 patterns. This corresponds to essence problems of finding the maximal stable set in a stable graph with the given number of nodes. We perform random-search hybridization for each of these problems, setting the pattern sizes to 2, 4, 8 and 16. Compared to the royal road problem definition in [20] which uses a chromosome of length 64, the hardest problem we tackle has a significantly larger search space: $2^{16 \cdot 16} = (2^{64})^4$.

All experiments carried out used for the main GA a population of 500 individuals and a rank based selection. Regardless of the setup, the same number of chromosome evaluations is performed and the count of offspring obtained via genetic operators is the same (although which operators get actually applied may differ due to the adaptive processes involved). Each run was halted after 10000 generations were completed or when the optimal solution was found, whichever came first.

## 4.2 Genetic operator types

We use two types of genetic operators, called *mutation-flex*$^+$ and *crossover-flex*.

For a royal road problem with $N$ bits, the *mutation-flex*$^+$ operator has $2 \cdot N + 1$ parameters. The first parameter, $a$, controls how many of the loci are considered for mutation. For each locus $i \in 1 \dots N$ we consider two parameters $p^i_{01}$ and $p^i_{10}$. Parameter $p^i_{01}$ gives the probability that the gene in the locus $i$ considered for mutation is flipped into 1, if its value is 0. Parameter $p^i_{01}$ has the dual meaning. This operator has the ability of discovering the "perfect" mutation for this problem: mutating each gene in a parent to its correct value in the offspring. However this ability is only potential, since the operator has no a priori knowledge on how this can be achieved. It is the responsibility of the adaptive system to tune the parameter values according to the experience derived from actual search space exploration.

The operator type *crossover-flex* has one parameter, the number of cutting points. We do not make any assumption on what is the appropriate number of cutting points (between 1 and $N - 1$) and let the adaptive system reach this decision.

## 4.3 "Learn As You Go" with IAGA$^+$

The big number of parameters of the *mutation-flex*$^+$ operator type creates bottlenecks when used with IAGA. Our experiments have shown that is difficult for the OAS to adapt this operator effectively, because the associated parameter search space is relatively vast compared to the experience an operator can earn. This lead us to evince a crucial knowledge which can also be extracted during the optimization process: namely *how* (or *why*) improvements and worsenings occur.

On top of the general mechanism of IAGA, we add a component which aims at tuning the operators *as they perform* their task of breeding new chromosomes in the main GA; it can be seen as a kind of lamarckianism in operator behavior evolution. The purpose of this extra-tuning is twofold: learn how to avoid making the same mistakes again and again and learn how to make yet more improvements from past positive experience. We call this type of tuning the behavior of operators *learn-as-you-go*. IAGA with a learn-as-you-go component is denoted by IAGA$^+$.

This idea may be seen as similar in spirit to the overall strategy of self-adaptive systems, but the two differ in a subtle way. In self-adaptive systems, each chromosome retains along with its genes, some additional information as to how operators should interact with it. This idea allows the individual to learn how to let operators modify it in order to increase its survival chances. Whenever an individual survives or passes genes, it also transmits its additional data (which consists of operator hints) to its offspring. In contrast to this, our proposal suggests that this type of knowledge should be kept within the operators, not the chromosomes. The first reason for this is that such a global knowledge repository accounts for a wide range of experience, which concerns the operator, not one chromosome. The second reason has to do with the importance of negative experience. Suppose that, in a self-adaptive system, one operator creates a considerably worse offspring. This negative information will be of course recorded in the additional data carried by that offspring, but, due to the selection mechanism involved, it may simply get lost. Thus, when operating on the same type of parent as the one lost, that operator may make the same mistake, again and again, simply because the negative information either disappears or is not reused.

We argue that by centralizing within the operator the knowledge regarding each operator's experience and effectiveness, the GA can make more effective use of such knowledge. Whenever this approach is practical for an arbitrary operator type $t$, we denote the corresponding learn-as-you-go system by LAYG($t$).

We illustrate this approach with LAYG(*mutation-flex*$^+$) for the royal road problem. Let us consider a particular *mutation-flex*$^+$ instance which has the settings $(p^i_{01}, p^i_{10})$, where $i \in \{1 \dots n\}$. At a given time, when this operator is applied and yields the offspring chromosome $c' = (b'_1 \dots b'_n)$ from the parent chromosome $c = (b_1 \dots b_n)$, by altering the bits with indexes $i_1 \dots i_l$, where $l \leq n$.

Whenever an (absolute) improvement is obtained (i.e. $c'$ has a greater fitness value than $c$), the parameter values of *mutation-flex*$^+$ are so that similar changes of the modified bits become more probable in the future:

**for all** $j \in \{i_1 \dots i_l\}$ **do**
  **if** $b_j = 0$ **then**
    increase $p^j_{01}$
  **else if** $b_j = 1$ **then**
    increase $p^j_{10}$

Of course, whenever a worsening occurs, an analogous algorithm is applied; in this case the corresponding probabilities are decreased, making such changes less probable in the future. Note how tuning adds to the effect of OAS.

There is a very interesting twist to this setup, related to plateau walks. We initially considered plateau walks to be neutral to this system. Therefore, when we took into account only improvements and worsenings, the mutation operators tended to converge to do-nothing ones, leading to deceiving results. This is caused by the fact that worsenings are much more frequent than improvements, and consequently the operators quickly learn that they are not supposed to do damage. But in this context, obtaining a plateau walk is a better thing than obtaining a worsening, which lead us to the idea of reinforcing plateau walks as well.

With LAYG in place however, one may wonder whether the original OAS in IAGA becomes redundant or ineffective, since on one hand it also aims at improving operators,

| Number of patterns: 4 | | | |
|---|---|---|---|
| pattern size | TGA | IAGA | IAGA$^+$ |
| 2 | 0 | 0 | 0 |
| 4 | 6.2 | 1.3 | 1.3 |
| 8 | 213 | 42.3 | 8.7 |
| 16 | — | 3465 | 1214.6 |
| Number of patterns: 8 | | | |
| pattern size | TGA | IAGA | IAGA$^+$ |
| 2 | 3.6 | 1 | 1.5 |
| 4 | 76.9 | 3.9 | 5.6 |
| 8 | 1442.2 | 216.7 | 68.4 |
| 16 | — | 8329 | 1360.7 |
| Number of patterns: 16 | | | |
| pattern size | TGA | IAGA | IAGA$^+$ |
| 2 | 17.6 | 3.4 | 3.9 |
| 4 | 304.5 | 42.2 | 30.6 |
| 8 | — | 857.4 | 258.3 |
| 16 | — | — | 2537 |

**Table 1: Average number of generations needed to reach the solution for GA variants ("—" means that no solution was ever found before 10000 generations). The number of patterns represents the number of nodes in the essence graph, which accounts for intrinsic combinatorial difficulty. Pattern size accounts for random-search hybridization of the essence problem.**

but on the other hand, it has fewer clues on how to do this effectively. We argue that the original OAS is to be kept, since it becomes a very important tool of sharing knowledge between operator variants.

### 4.4 Comparative results for various levels of random-search hybridization

Table 1 gives the experimental results obtained by the traditional GA (TGA), IAGA and IAGA$^+$ for the royal road problem. We evince that adaptation is material in tackling random-search induced difficulty on the same essence problem.

The traditional GA has been carefully tuned to achieve a good performance. We used one-point crossover with a rate of 0.5 and a mutation operator with a rate of 0.3; when applied, the mutation operator changes a gene's value with 0.025 probability. These settings may look arbitrary, but this very fact evinces a shortcoming of this approach when compared to adaptive systems. Nevertheless, our experiments have shown that further tuning of these parameters do not change the results in a fundamental way.

IAGA adds to TGA the RAS and OAS (and thus eliminates the need for parameter tuning). IAGA$^+$ adds to IAGA a learn-as-you go system for the *mutation-flex*$^+$ operator.

When the random-search hybridization is low (pattern size is 2 or 4) there are little differences between the three approaches. We can even see that for the easiest problem (4 patterns of size 2) the process of randomly generating a population is enough to give rise to the solution. As the random-search hybridization increases, the situation changes radically.

The population size (which is the same for all experiments) is no longer a factor which, by sheer exploration, can produce a solution very rapidly. Instead, hundreds of generations or more are needed by the traditional GA to reach the solution. When the random-search hybridization prescribes a pattern size of 8, the contribution of the adaptive systems becomes significant. For the hardest problem which defines 16 patterns random-search hybridized to size 16, it is only IAGA$^+$ (that does *not* "find and forget", nor "destroy and forget") which manages to find a solution.

## 5. CONCLUSIONS AND FUTURE WORK

Adaptation is the basic keyword of Holland's pioneering textbook on GAs. The GAs evolve solutions through adaptation of candidate solutions. The idea that elements of the GA should also adapt is more and more widespread. IAGA defines a framework which adds two elements to the standard GA scheme: rate adaptive system and operator adaptive system. IAGA$^+$ builds on IAGA providing a learn-as-you-go module which allow operators to directly adjust themselves based on the results of their application.

Experimental results show that simply having adaptive elements in a GA may not be sufficient. It is important to design adaptive elements which have ability to adapt in a way that suits the optimization process. When this ability becomes too general, a greater degree of exploitation of the search experience is needed. IAGA$^+$ addresses this need, showing significant performance improvements over IAGA and traditional GA. It is likely that a conclusion can be stated as: "The more relevant adaptation included in the GA, the more problems can be actually solved".

Our study may prove relevant for other problems which can be reduced to an NK fitness landscape. This is not to imply that precisely the design of the operators we used is appropriate for an arbitrary problem (which, for example, may use chromosomes which encode integer or floating point values). However, the framework provided by IAGA$^+$ can be applied on top of a traditional GA by creating custom components specific for the problem at hand.

In this paper we investigated only the effect of random-search hybridization on NK fitness landscapes which exhibit no pattern conflicts. We intend to establish whether IAGA$^+$ produces the same positive effects in tackling random-search hybridized on problems with greater intrinsic difficulty. Another direction is to study the effect of an integrated population size control mechanism.

## 6. REFERENCES


[1] L. Altenberg. *Handbook of Evolutionary Computation*, chapter B2.7.2 NK Fitness Landscapes. 1997.

[2] T. Bäck. Optimal Mutation Rates in Genetic Search. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, 1993.

[3] H.-G. Beyer and K. Deb. On the desired behaviors of self-adaptive evolutionary algorithms. In *Parallel*

*Problem Solving from Nature – PPSN VI*, pages 59–68, 2000.

[4] A. E. Eiben and T. Back. Empirical investigation of multiparent recombination operators in evolution strategies. *Evolutionary Computation*, 5(3):347–365, 1997.

[5] A. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Trans. on Evolutionary Computation*, 3(2):124–141, 1999.

[6] A. E. Eiben, E. Marchiori, and V. Valko. Evolutionary algorithms with on-the-fly population size adjustment. In *Parallel Problem Solving from Nature*, volume LNCS, pages 41–50, 2004.

[7] M. R. Garey and D. S. Johnson. *Computers and Intractability:A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co, 1979.

[8] D. E. Goldberg. Sizing populations for serial and parallel genetic algorithms. In J. Schaffer, editor, *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 70–79, San Francisco, CA, 1989. Morgan Kaufman.

[9] D. E. Goldberg, K. Deb, and J. H. Clark. Genetic Algorithms, Noise, and the Sizing of Populations. IlliGAL Report 91010, University of Illinois at Urbana-Champaign, 1991.

[10] G. Greenwood. Finding solutions to NP problems: Philosophical differences between quantum and evolutionary search algorithms. In *Congress on Evolutionary Computation*, pages 815–822, 2001.

[11] M. Grotschel, L. Lovasz, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, (1):169–197, 1981.

[12] G. R. Harik and F. G. Lobo. A parameter-less genetic algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pages 258–265, 1999.

[13] R. Hinterding, Z. Michalewicz, and T. C. Peachey. Self–adaptive genetic algorithm for numeric functions. In *Parallel Problem Solving from Nature – PPSN IV*, pages 420–429, 1996.

[14] B. A. Julstrom. What have you done for me lately? Adapting operator probabilities in a steady-state genetic algorithm. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 81–87, 1995.

[15] S. A. Kauffman. *The origins of order. Self-Organization and selection in evolution*. Oxford University Press, 1993.

[16] F. G. Lobo, K. Deb, D. E. Goldberg, G. R. Harik, and L. Wang. Compressed introns in a linkage learning genetic algorithm. In *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 551–558, 22-25 1998.

[17] H. Luchian and O. Gheorghies. Integrated-adaptive genetic algorithms. In *ECAL*, pages 635–642, 2003.

[18] Z. Michalewicz and D. Fogel. *How to Solve It: Modern Heuristics*. Springer, 2002.

[19] G. Minty. On maximal independent sets of vertices in claw free graphs. *Journal of Combinatorial Theory*, Ser. B(28):284–304, 1980.

[20] M. Mitchell, S. Forrest, and J. H. Holland. The royal road for genetic algorithms: Fitness landscapes and GA performance. In F. J. Varela and P. Bourgine, editors, *Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life, 1991*, pages 245–254, Paris, 11–13 1992. A Bradford book, The MIT Press.

[21] C. Reeves. Using genetic algorithms with small populations. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 92–99, San Mateo, CA, 1993. Morgan Kaufman.

[22] J. Smith and T. C. Fogarty. Operator and parameter adaptation in genetic algorithms. *Soft Comput.*, 1(2):81–87, 1997.

[23] D. Thierens. Adaptive mutation rate control schemes in genetic algorithms. In D. B. Fogel, M. A. El-Sharkawi, X. Yao, G. Greenwood, H. Iba, P. Marrow, and M. Shackleton, editors, *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*, pages 980–985. IEEE Press, 2002.

[24] E. D. Weinberger. NP completeness of Kauffman's N-k model, a tuneably rugged fitness landscape.