ELSEVIER

Theoretical Computer Science 287 (2002) 167–185

Theoretical Computer Science

www.elsevier.com/locate/tcs

GA performance distributions and randomly generated binary constraint satisfaction problems

B. Naudts*,[1], L. Schoofs

Department of Mathematics and Computer Science, University of Antwerpen (RUCA), Groenenborgerlaan 171, B-2020 Antwerpen, Belgium

Abstract

We investigate the variable performance of a genetic algorithm (GA) on randomly generated binary constraint satisfaction problem instances which occur near the phase transition from soluble to non-soluble. We first carry out a conventional landscape analysis and observe, next to a number of common features related to the interaction structure, important differences between the instances, such as the number of solutions, the quality of the paths to the solutions, and the lengths and extent of the neutral paths for mutation. We then split the dynamics of the GA into two phases: the ascent towards the high fitness region, and from this high fitness region to a solution. To gain further information about the GA's behavior in the first phase, we construct two models based on the much simpler fully separable functions, and try to match instances which show a similar performance distribution. Although far from exact, this technique of comparing with analog search problems gives a hint about the landscape elements that are responsible for the GA's slow ascent.

Keywords: Genetic algorithms; Constraint satisfaction problems; Landscape analysis; Performance distribution

1. Introduction

Binary constraint satisfaction problems (CSPs, e.g. [19]) form an NP-complete class of search problems rooted in operations research. Interest in randomly generated instances arose in the early 1990s when a phase transition from solvable to unsolvable was discovered when varying parameters of instance generators of several NP-complete problems [3]. Instances generated near the phase transition proved to be particularly

* Corresponding author.

E-mail addresses: bart.naudts@ua.ac.be (B. Naudts), luk.schoofs@ua.ac.be (L. Schoofs).

[1] Post doctoral researcher of the Fund for Scientific Research, Flanders Belgium (F.W.O.-Vlaanderen).

0304-3975/02/$ - see front matter
PII: S0304-3975(02)00133-0

hard to solve, much harder than the instances away from it. Such a phase transition was shown to occur in several random instance generators of the binary CSP [22]. This turns them into useful, tunable problem generators where the hardness of a problem can be controlled by only one or a few parameters.

Genetic algorithms (GAs, [10,7,2]) are stochastic search algorithms based on the principle of 'survival of the fittest'. They have been applied to a wide variety of optimization and search problems, partly because of their very good implementation-cost-to-performance ratio. In applications, they usually do not appear in their basic form called the simple GA, but in a form specialized towards the problem to be solved, often incorporating additional heuristics. The GA has been specialized in many different ways to solve CSPs; we refer to [5,4] for an overview.

The dynamics of a simple GA operating on fully separable problems, i.e., search problems where the variables do not interact with each other but provide independent contributions to the fitness value, has been studied analytically and in great detail using both dynamical systems [20] and statistical physics [17] approaches. Consisting only of second-order interactions, one (major) step beyond full separability, the class of binary CSPs has up to now proved to be too rich and difficult to tackle with these techniques. An alternative approach to learn about how a GA behaves on harder search problems is to construct problem instance generators to analyze the instances *statically*, i.e., not within the dynamics of a GA (see e.g. [12] for an overview of techniques), and to empirically study the difference in GA dynamics when varying the parameter(s) of the generator. The NK-landscape [13] is the most prominent example of such generators.

The first part of this contribution contains a conventional static analysis of CSP instances near the phase transition, based on hill-climber and random walk experiments for the mutation landscape, a Walsh analysis, and a schema analysis to catch some of the crossover and population effects. Unfortunately, the analysis provides no means for classifying the instances according to their GA performance, i.e., none of the information can on its own be used for reliable a priori performance prediction.

What we do learn is that there are two possible causes for the slow ascent of the GA toward the high fitness region: the high collateral noise between the low-order building blocks, and the existence of a number of attractors which distract the GA from a direct path to a solution.

The key idea explored in the second part of the paper is to learn from simpler search problems which we design to be similar to the CSP instances in many aspects. They share several static landscape properties, such as fitness distance correlation and schema fitness distribution overlaps. Concentrating on the lengths of the paths towards the high fitness region, we also require them to show a similar distribution of the number of generations needed by the GA to reach this region. We call this distribution the *performance distribution*.

By matching the performance distributions of a CSP instance to that of an instance of each of the two models, and designing each model to reflect one of the suspected causes of slowdown, we can answer the following questions:

- If collateral noise is the only cause of the slow ascent, how much evaluation noise would we have to add to a fully separable function to achieve the same slowdown?
- If not noise but different attractors causing meta-stable states are the main cause,

how many of these attractors would we need, and how strong would they have to be, to cause the same slowdown?

The remainder of this paper is organized as follows. We define the CSP instance generator in Section 2, and describe the GA and classify its observed performance on a set of 100 instances in Section 3. In Section 4 we present the static landscape analysis; a summary of the analysis can be found in Section 4.3. Section 5 is devoted to study of the ascent of the GA by comparing model instances which show similar performance distributions. We end the paper with Section 6, presenting conclusions and an overview of further work. Appendix A contains a short overview of the Walsh transform. Appendix B contains a direct calculation of the Walsh coefficients for binary CSPs.

2. Binary constraint satisfaction problems

A binary constraint satisfaction problem can be defined as

- a set of variables x_i with $i \in L = \{0, \ldots, \ell - 1\}$;
- a domain or alphabet $\Sigma = \{0, \ldots, n - 1\}$;
- a subset $C_{ij} \subset \Sigma \times \Sigma$ for each pair of variables (i, j), with $0 \leqslant i < j < \ell$, which represents a *constraint* when it differs from the Cartesian product $\Sigma \times \Sigma$. The elements of $\Sigma \times \Sigma \backslash C_{ij}$ are called *conflicts*.

We refer to [19] for more general definitions, and drop the predicate 'binary' to solely use the abbreviation CSP.

The goal of a CSP is to assign to the variables x_i a value from their domain Σ in such a way that all constraints are satisfied. Formally, we say that a constraint C_{ij} is satisfied if and only if $(x_i, x_j) \in C_{ij}$. The couple (x_i, x_j) is then called a *valid assignment*. When $(x_i, x_j) \notin C_{ij}$ we say that the assignment (x_i, x_j) *violates* the constraint C_{ij}. The action taken by an algorithm to check if a constraint is violated or not by an assignment is called a *constraint check*. The terms *candidate solution*, *individual* and *string* both refer to a particular assignment of values to the variables.

CSPs do not have an explicit cost function associated to them. Black-box algorithms like the GA require one because they cannot evaluate partial assignments. In the context of randomly generated instances, the *standard penalty function*, which counts the number of violated constraints, is the most sensible one to use. We will use it throughout the paper and call it the fitness function, sticking to GA terminology.

There exist different models for randomly generating CSP instances (see [14] for an overview). We use the one-parameter *model E* [1]: *Select uniformly, independently and with repetition, $pn^2\ell(\ell-1)/2$ conflicts out of the $n^2\ell(\ell-1)/2$ possible*. In the limit of the number of variables going to infinity, a phase transition occurs from solvable to unsolvable when varying p. When the number of variables is finite, a parameter region exists where the expected number of solutions is low, and solvable and unsolvable instances coexist. It is called the *mushy region*. For an in-depth discussion about the structures generated with this model, and its theoretical background, we refer to [14,1].

The experimental results presented in this paper are based on 100 solvable instances of 30 variables and a domain size of 5, generated with $p = 0.099$, which is at the very

end of the mushy region. Only three of these instances can be solved within 50,000 constraint checks of a specialized forward-checker with conflict-directed back-jumping [16]. The motivation for selecting only solvable instances is one of convenience; we went as far as possible into the mushy region to avoid almost flat instances with a very large amount of solutions.

3. Performance of the GA

The simple GA whose performance on CSP instances we have tested is a generational GA with a population of size 50, binary tournament selection, per-bit mutation at a rate of $1/\ell \approx 0.033$, and one-point crossover with probability 0.8. The values of the parameters were chosen because they are fairly standard and they yield reasonable performance. The only parameter we will vary in the experiments described in this paper is the crossover rate.

One immediately observes that there are easy and hard instances. We will use the following loose classification based on 20 independent GA runs per instance:

(1) *easy*: the median number of generations required to find a solution is less than 3000 (25 instances);
(2) *medium hard*: the median number of generations required to find a solution is between 3000 and 12,000 (48 instances);
(3) *hard*: the median number of generations required to find a solution is higher than 12,000 (27 instances).

The ultimate aim of a static landscape analysis is to rank instances according to their GA performance by only analyzing features of the problem instance. (This process is also called a priori problem difficulty prediction.) The landscape analysis we carry out in the next section will not provide sufficient information to do this reliably, which is why we have not chosen to classify the instances more stringently.

4. Static landscape analysis

In this section, we present a static landscape analysis of the 100 instances. Since most of the techniques we applied are only qualitatively reliable (e.g., a difference of 0.5 in the fitness distance correlation is likely to be of importance, but a difference of 0.1 may not be), we will only summarize the results of each analysis.

4.1. The mutation landscape

We started the analysis of the mutation landscape by performing 100 independent hill-climber[2] runs on each instance. The number of times it cannot find a solution is low for easy instances (mean 41.3, std. dev. 16.5), high for hard instances (mean 86.8, std. dev. 7.2), and in between (mean 64.3, std. dev. 15.1) for medium hard

[2] The hill-climber modifies the value of one variable per iteration.

instances. These results indicate that the mutation landscape plays a significant role, but it is not all-dominant, in the sense that hill-climber performance cannot be used for an individual classification of GA performance. The distributions of hill-climber performance per category overlap significantly: the minimum (26) and the maximum (89) of the 'medium hard' category are both beyond the means of the other classes.

As a by-product of the hill-climber runs we recorded the number of times a neutral step, i.e., one which does not change the fitness, was taken. This number is largely instance and run independent, and varies between 0.02 and 0.06. Given that there are 30 variables and 5 alphabet elements, i.e., 149 possible mutations, we found an average of about 6 possible neutral steps per individual.

Next, we performed, for each instance, a random walk of 250,000 steps starting from a solution, restricted to individuals in the fitness range 0–3 (i.e., mutations which led to inferior individuals were ignored). We recorded the number of solutions encountered during the walk. Easy instances have a high recorded number, hard instances a low one, but because high and low numbers occur in the medium hard category we can again not use this number to classify instances individually. We can only conclude that easy instances have more solutions, hard instances have fewer, and that the landscape is flat enough for many solutions to be interconnected by fitness 0–3 paths.

The restriction on the order of the interactions between the variables to pairwise interactions ensures that the landscape is smooth in the sense that the fitness values of individuals in the near Hamming neighborhood can be well predicted. This is mathematically formalized by the *correlation length* (first applied in a GA context in [15]), which we have experimentally verified to be around 10 for all instances. Note also that the isotropy condition on the landscape, required to compute the correlation length using random walks, is satisfied because of the random generation of the conflicts of the CSP. The correlation lengths obtained by starting the random walks in sub-optimal individuals therefore yield identical values; the structure of the mutation landscape is similar in average fitness and high fitness regions.

The *fitness distance correlation* [11] of a mutation landscape is a measure of the extent to which the fitness values of candidate solutions correlate with their Hamming distance to a solution. It is computed using the formula

$$\mathrm{fdc}_S(f) = \frac{\sum_{s\in S}(f(s)-\bar{f}_S)(d(s,s^*)-\bar{d}_S)}{(\sum_{s\in S}(f(s)-\bar{f}_S)^2)^{1/2}(\sum_{s\in S}(d(s,s^*)-\bar{d}_S)^2)^{1/2}},$$

where S denotes a sample of the search space, d the Hamming distance metric, f the fitness function, s^* a solution, and

$$\bar{f}_S = \frac{1}{|S|}\sum_{s\in S} f(s) \quad \text{and} \quad \bar{d}_S = \frac{1}{|S|}\sum_{s\in S} d(s,s^*).$$

The *onemax* problem, which counts the number of ones in a binary string, is an example of a problem which shows a perfect correlation. We computed the measure based on a sample of the search space near a solution, and found good correlation values between 0.7 and 0.8, independently of the choice of solution and instance.

Finally, we performed a Walsh (discrete Fourier) transformation (see Appendix A, and [6] for a discussion in the context of GAs) of the instances, based on the formulae presented in Appendix B. Note that all Walsh coefficients of order 3 and higher are necessarily zero due to the binary interaction structure of the problem. A first observation is that the magnitudes of the first-order coefficients are all very similar, irrespective of the instance: their modulus is largely within the range 0.1–0.6. The same holds for the second-order coefficients, whose modulus lies between 0.02 and 0.09. This similarity will be discussed in the context of schema fitness distributions, in the next section.

To measure the importance of the first- versus second-order Walsh coefficients, we split the fitness value in a constant term (which is by far the largest), a first-order contribution and a second-order contribution:

$$f(s) = \sum_{j=0}^{n'} \omega_j \psi_j(s) = \omega_0 + \sum_{\text{1st order } j} \omega_j \psi_j(s) + \sum_{\text{2nd order } j} \omega_j \psi_j(s)$$

and observe, by randomly generating and evaluating, that in the average fitness region, the second-order contribution is larger than the first-order in about 59% of the individuals. On average, however, the first-order contribution is a factor 2 larger than the second-order. We conclude that the instances contain an important 'fully separable' or first-order component (whence the low fitness distance correlation), and that the second-order component plays an equally strong role (it generates the local optima in the mutation landscape). We also observed that the standard deviation of the size of the first-order contributions has a similar classification power as the hill-climber runs and the random walks in the high fitness region. Statistics can be found in Table 1.

4.2. A schema analysis

We start with some terminology. A *schema* is a hyperplane of the search space Σ^L. It is usually written as a string over the augmented alphabet $\Sigma' = \Sigma \cup \{*\}$, where the $*$ plays the role of a wild card symbol. Technically, an individual $s_0 \ldots s_{\ell-1}$ belongs to a schema $h_0 \ldots h_{\ell-1}$ if $s_i = h_i$ for all i such that $h_i \neq *$. We use the term *schema fitness distribution* to denote the distribution of fitness values of all individuals belonging to a schema. The *order* of a schema is given by the number of non-wild card symbols in the schema, the *length* of a schema is given by the largest distance between two non-wild-card symbols. A *hyperplane partition* consists of all schemas with wild-card symbols on the same positions. A *schema competition* is defined as the comparison of the average fitness values of all schemas in a hyperplane partition.

Table 1
Statistics of the standard deviation on the size of the first-order contribution in the fitness function of CSP instances, shown per category of GA performance

Category	Mean	Standard deviation	Minimum	Maximum
Easy	3.61	0.22	3.28	4.06
Medium hard	3.57	0.24	3.07	3.94
Hard	3.47	0.27	2.99	4.06

The concepts of a schema and its fitness distribution are central to a simple and intuitive hypothesis about the dynamics of a GA, which we present in the form of the *static building block hypothesis* [9]:

Given any short, low-order hyperplane partition, a GA is expected to converge to the winner of the corresponding schema competition.

Fit, short, low-order schemas are called *building blocks*. Note that the word *static* is again used to denote that no actual GA dynamics is involved. A schema is called deceptive when (a) it is the winner of its schema competition and (b) no solution is contained in it. A search problem is called deceptive when it contains deceptive low-order schemas. We refer to [21] for a discussion of deception. According to the hypothesis, deceptive problems should be hard for a GA because they mislead its search for a solution.

We have experimentally verified the correlation between deception and GA performance on the 100 instances, based on all the optima we found for each instance. The results are similar to those of previous experiments: easy instances contain few deceptive schema competitions, hard instances contain many, but the middle group spans the whole range. Deception cannot be used for the classification of individual instances; the following paragraphs give a hint of why this may be.

Irrespective of the instance and the order (1,2,3) of the hyperplane partition, we observe that both mean and standard deviation of the schema fitness distributions in a partition do not vary very much. Typical values for first- and second-order partitions are between 39 and 42 for the mean and between 5 and 7 for the standard deviation—clearly the distributions overlap significantly. In many schema competitions, the difference in mean between two schemas (the *signal*) is 2 orders of magnitude smaller than the average standard deviation (the (*collateral*) *noise*). An application of the population sizing rules of Goldberg et al. [8] would therefore lead to excessively large population sizes to allow schema competition winners to dominate.

Given that the population size of our GA is only 50, and that population sizes of at least 3 orders of magnitude larger would be necessary for the GA to decide between building blocks, we expect that deception (w.r.t. static schema fitness distributions) does not play a role of importance. Because of the large overlaps of the distributions, we conjecture that the amount of deception is directly linked to the number of solutions in the instance: the probability that a schema competition winner is contained in a solution increases with the number of solutions. Of course, there is a correlation between the number of solutions and GA performance, which gives deception a similar classification power.

The absence of strong schema competition winners makes it very unlikely that combinations of schema competition winners can be used to assemble a solution in linear time. Indeed, if we group all first-order winners, we obtain an individual which violates between 16 and 26 constraints, depending on the instance. Similar results hold for consensus individuals assembled with second-order winners. However, a quadratic time greedy algorithm which fixes variables successively based on first-order (w.r.t. the free variables) schema means reaches near-optimal values.

This observation, together with the fact that many GA runs reach the high fitness region in a reasonable number of generations, indicates that the signal-to-noise

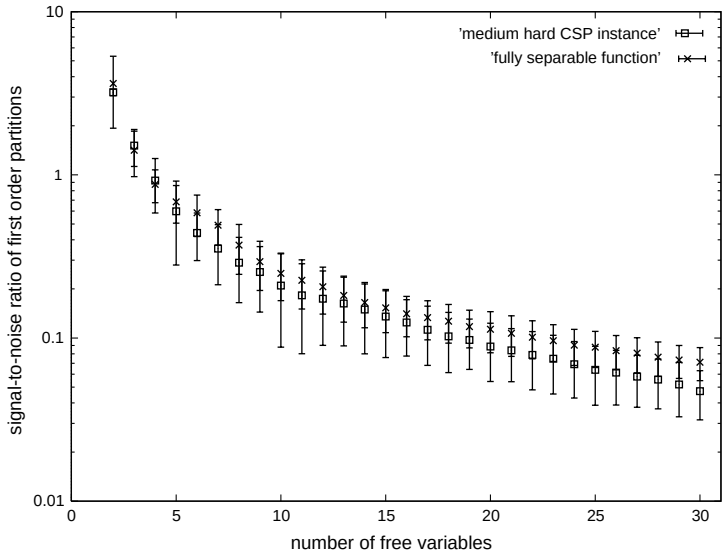

Fig. 1. The signal-to-noise ratio associated with the highest signal within a first-order partition (first-order w.r.t. the free variables), averaged over all first-order partitions, for an increasing number of free variables, starting from a solution. The data are based on a sample of 10,000 strings per partition, of one medium hard instance and a fully separable function, generated according to the noise model of Section 5.2 but without noise term.

ratio improves with the number of variables getting fixed. Fig. 1 confirms: it shows the signal-to-noise ratio associated with the highest signal within a first-order schema competition (first-order w.r.t. the free variables), averaged over all first-order competitions, for an increasing number of free variables, starting from a solution. The data are based on a sample of 10,000 strings per partition of one medium hard instance.

The figure also shows that the signal-to-noise ratios differ little from those of a fully separable function. This similarity can be explained for the region with many fixed variables as follows: each time a variable gets fixed, the second-order interaction between this variable and a free variable is converted into an independent (non-interacting) contribution to the free variable. These independent contributions help break the symmetry which is present when all variables are free, and make the function look more like a fully separable one. The fact that the signal-to-noise ratios improve in a fully separable function with the number of variables getting fixed is solely due to the reduced variance.

4.3. Summary

(1) There is a good but not perfect correlation between GA performance and the proportion of times a hill-climber gets stuck in a local optimum.

(2) The first- and second-order components play an equally important role. The second-order component is responsible for local optima, but it is most of the times not strong enough to catch the GA in a local optimum.
(3) The high number of possible neutral mutation steps ensures a good connectivity in the 0–3 fitness band, where the GA typically spends a large part of its time.
(4) The ascent toward high fitness is initially slowed down by the high amounts of collateral noise between building blocks. As search progresses, the GA is able to pick up a stronger signal.

5. Matching performance distributions

In this second part of the paper we focus on the ascent of the GA toward the high fitness region (the *first phase*). We study the structure of the CSP instances further by comparing two models on which the GA shows a similar performance distribution.

5.1. The technique

We define the *performance distribution* of the GA on a search problem as the distribution of number of generations required to find a solution (or to reach the high fitness region, depending on the object of interest), obtained over many independent runs. Fig. 2 shows 6 such distributions, plotted as histograms, for two different problems and each time three different crossover rates. Note that the runs are grouped according to the *logarithm* of the number of generations to get a better view on the short runs.

The performance distribution is a signature of the GA-search problem combination. It is well possible that different combinations yield similar distributions; however, as Fig. 2 shows, the differences can be large and the shape of the distribution may be retained under variations of GA parameters. Important characteristics of the search landscape are reflected in the distribution. Observe, for example, the similarity in shape of the needle-in-a-haystack problem in Fig. 2 and the second phase (from fitness 3 to a solution) of a GA with crossover on a medium hard CSP instance in Fig. 3.

The models presented in the next two sections are designed to have first-phase performance distributions which can match those of the CSP instances. They are also designed to have similar landscape characteristics:

- Their fitness distance correlation value is between 0.7 and 0.8.
- The overlap of the low-order schema fitness distributions within a competition is within the same order of magnitude. Stronger, the curves of signal-to-noise ratios of first-order schemas with respect to free variables match reasonably well.
- They contain an important first-order component, as they are based on fully separable functions.

5.2. The noise model

The first family of functions which we use to model the first phase of the GA is based on one fully separable function on which uniform noise is added. The functions

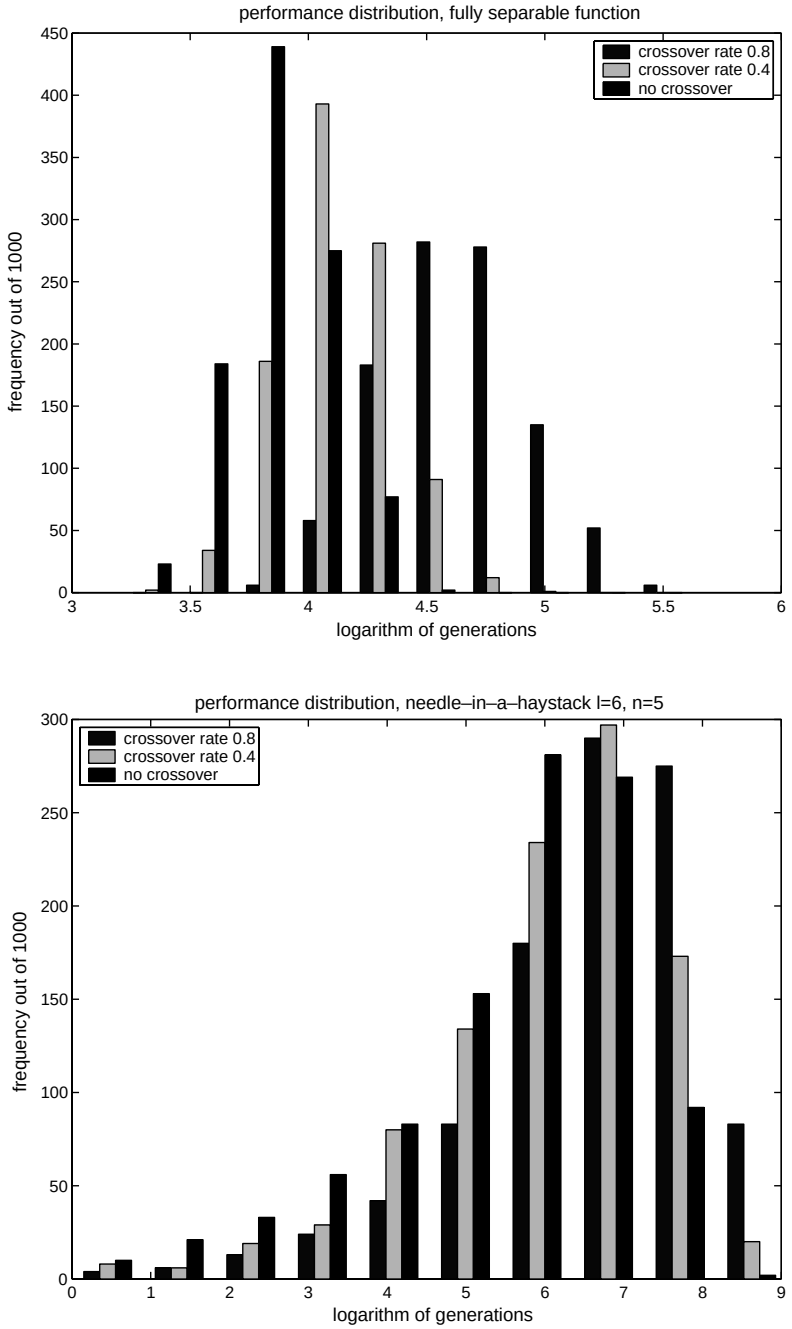

Fig. 2. Performance distributions of the GA on a fully separable function (again with 30 variables and 5 alphabet elements) and a 6-variable, 5 alphabet elements needle-in-a-haystack problem.

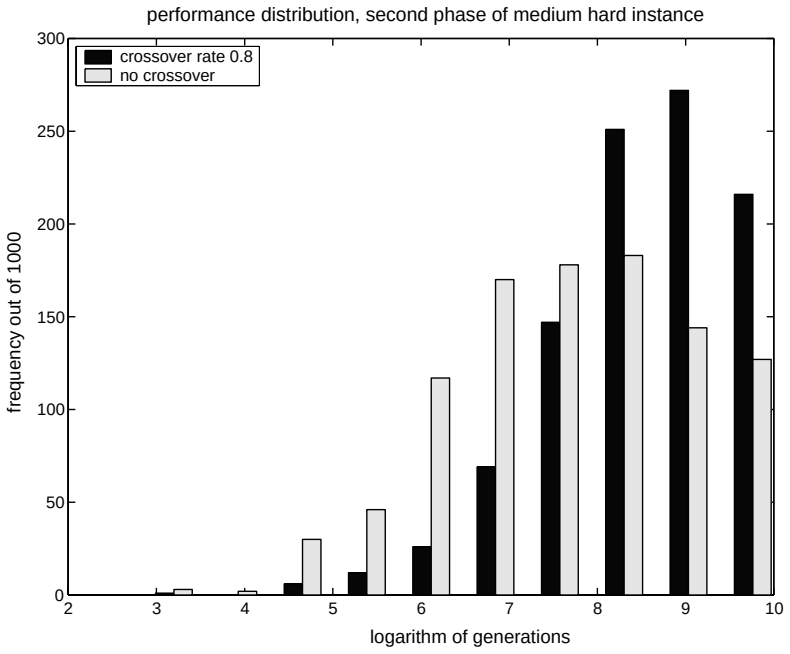

Fig. 3. Performance distributions of the second phase of the GA on a medium hard CSP instance, with crossover (rate 0.8) and without. Note the typical form of the distribution of the GA with crossover, which corresponds well to the form of the needle-in-a-haystack performance distributions.

are of the form

$$f(s) = \sum_{0 \leqslant i < \ell} g_i(s_i) + U(a,b), \quad \forall s \in \Sigma^{\ell}. \tag{1}$$

The values $g_i(0)$ are set to 0, the values $g_i(a)$, $a \neq 0$, are drawn from a uniform distribution on the set $\{1,2,3,4\}$. At each function evaluation, we add noise which is drawn from a uniform distribution on the set of integers $\{a,\ldots,b\}$, where a and b are the parameters of the family.

The motivation for this model is two-fold: we want to investigate whether the second-order components can be replaced by a noise term, and we want to find out whether the small difference between the signal-to-noise ratio of the fully separable function and that of the CSP instance is important. Adding the (fitness independent) noise to the fully separable function decreases the signal-to-noise ratio appropriately in the region of many free variables, as shown in Fig. 5, at the expense of a worse match when the number of fixed variables is high.

Fig. 4 shows the conformance w.r.t. performance distribution of the model for an easy, medium hard and hard instance. The noise terms we used are respectively $U(-4,9)$, $U(-2,11)$ and $U(-2,15)$, which shows that we can slow down the GA quite a bit by almost systematically underestimating the fitness of the individuals.

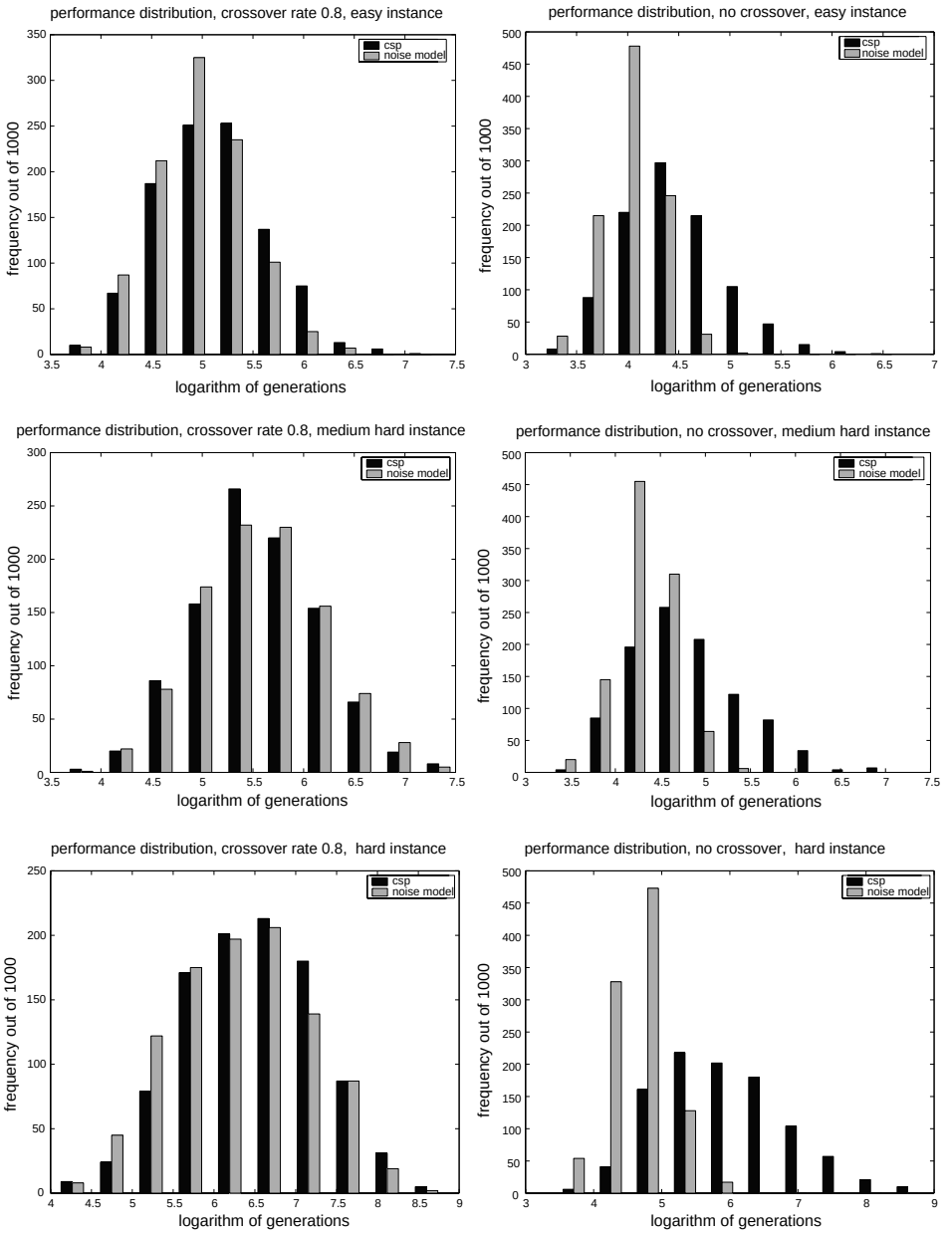

Fig. 4. Performance distributions of the GA on three CSP instances (rows: easy, medium hard, hard) and instances of the noise model which match for a crossover rate of 0.8 (first column). When crossover is switched off (second column), the distributions do not match anymore. Parameter settings for the model are detailed in Section 5.2.

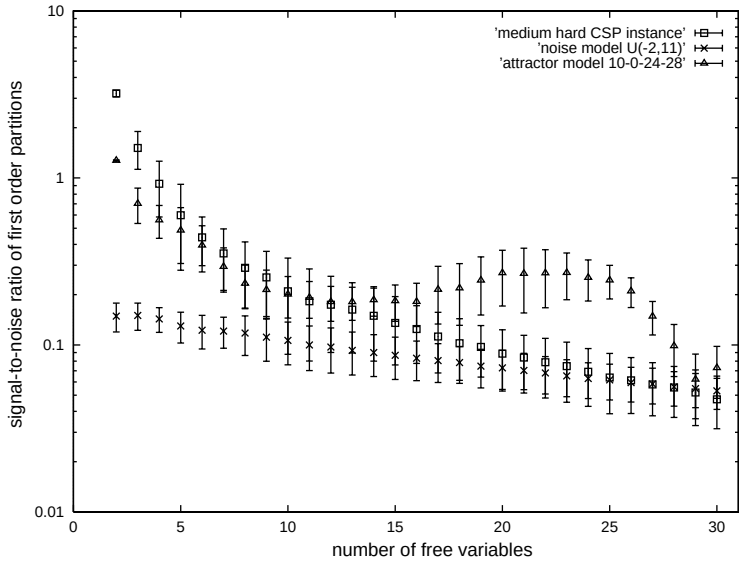

Fig. 5. The signal-to-noise ratio associated with the highest signal within a first-order schema competition (first-order w.r.t. the free variables), averaged over all first-order competitions, for an increasing number of free variables, starting from a solution. The data are based on the same medium hard instance as in Fig. 1, and the corresponding noise and attractor model.

The main problem with this model is that it appears to be unstable with respect to crossover rate variation, i.e., if we fix the model parameters which caused a good match for a GA with crossover rate 0.8, we find that the distributions do not match anymore when the crossover rate is decreased to 0.4 or crossover is switched off.

5.3. The attractor model

A second parametrized family of functions used to model the first phase of the GA running on the CSP instances is based on a combination of fully separable functions, each associated with an attractor string. Here, we assume that the differences in signal-to-noise ratios are not that important, but that the slowdown is caused by the presence of attractors that make the GA lose time.

The primary attractor t^0 is the all 0 string. We also use K secondary attractors t^k, $k=1\ldots K$. With probability ξ, $t_i^k=0$. Otherwise it is assigned a value drawn from a uniform distribution on $\{1\ldots4\}$. The parameter ξ controls the Hamming distance to the primary attractor.

The joint fitness function is defined as the minimum of the fitness functions f'^k associated with attractor t^k, $0\leqslant k\leqslant K$. Each of these will make use of the same set of penalties, defined as follows. Let for all $a\in\Sigma$, $g_i(a)=1$ with probability 0.5. Otherwise, we assign a number drawn from a uniform distribution on $\{2,3,4,5\}$. This system of

assignments ensures a fitness distance correlation which is similar to that of the CSP instances.

The fitness functions f^k associated to the attractors are based on functions h^k

$$h^k(s) = \sum_{0 \leqslant i < \ell} z(k,i,s_i) \quad \text{with } z(k,i,s_i) = \begin{cases} 0 & \text{when } s_i = t_i^k, \\ g_i(s_i) & \text{otherwise.} \end{cases} \tag{2}$$

The main idea here is that penalties are given for each position where an individual deviates from the attractor. The primary attractor f^0 is defined to be exactly h^0. To avoid $K+1$ optima, we modify the value of h^k, $k>0$, when an individual comes too close to the attractor:

$$f^k(s) = \begin{cases} h^k(s) & \text{when } d(s,t^k) > P_1, \\ \max(h^k(s), P_2) & \text{otherwise,} \end{cases} \tag{3}$$

where $d(\cdot,\cdot)$ denotes the Hamming distance between two strings. This modification turns the upper part of the basin of attraction into a plateau, which will cause a hill-climber or a GA to loose time by going to the plateau. However, when P_1 and ξ are sufficiently large, it does not create a local optimum in which the algorithms get stuck.

Fig. 6 shows the conformance of the model for the same easy, medium hard and hard instance as used in Fig. 4. The parameter settings of the model are detailed in Table 2. It proved harder to find parameters which produced a good match, especially with the long tails in the distribution of the model. In contrast to the noise model, the attractor model is reasonably stable with respect to crossover rate variation. The signal-to-noise ratio versus the number of free variables plot for the (10,0,24,28) model instance, on the other hand, is very different from that of the medium hard CSP instance.

5.4. Discussion

Rigorously speaking, we have only answered the following two questions:

- How much evaluation noise do we have to add to a fully separable function to achieve the same slow ascent toward the high fitness region as a GA does on a CSP instance?
- How many 'fully separable' attractors do we need, and how strong do they have to be, to cause the same slowdown?

for *one particular choice of algorithm*, i.e., one choice of parameter setting of the GA. We observed that the attractor model is relatively stable with respect to crossover rate variation. For the noise model, however, we need different model instances for different GA parameter settings.

Of course, similarity of two search problems based on their performance distributions does not imply that the problems are identical to a GA. It is probable that very weird search problems can be constructed which still show a similar performance distribution to that of the CSP instances. We neither have any guarantee that the models represent important aspects of the low fitness region of the CSP instances.

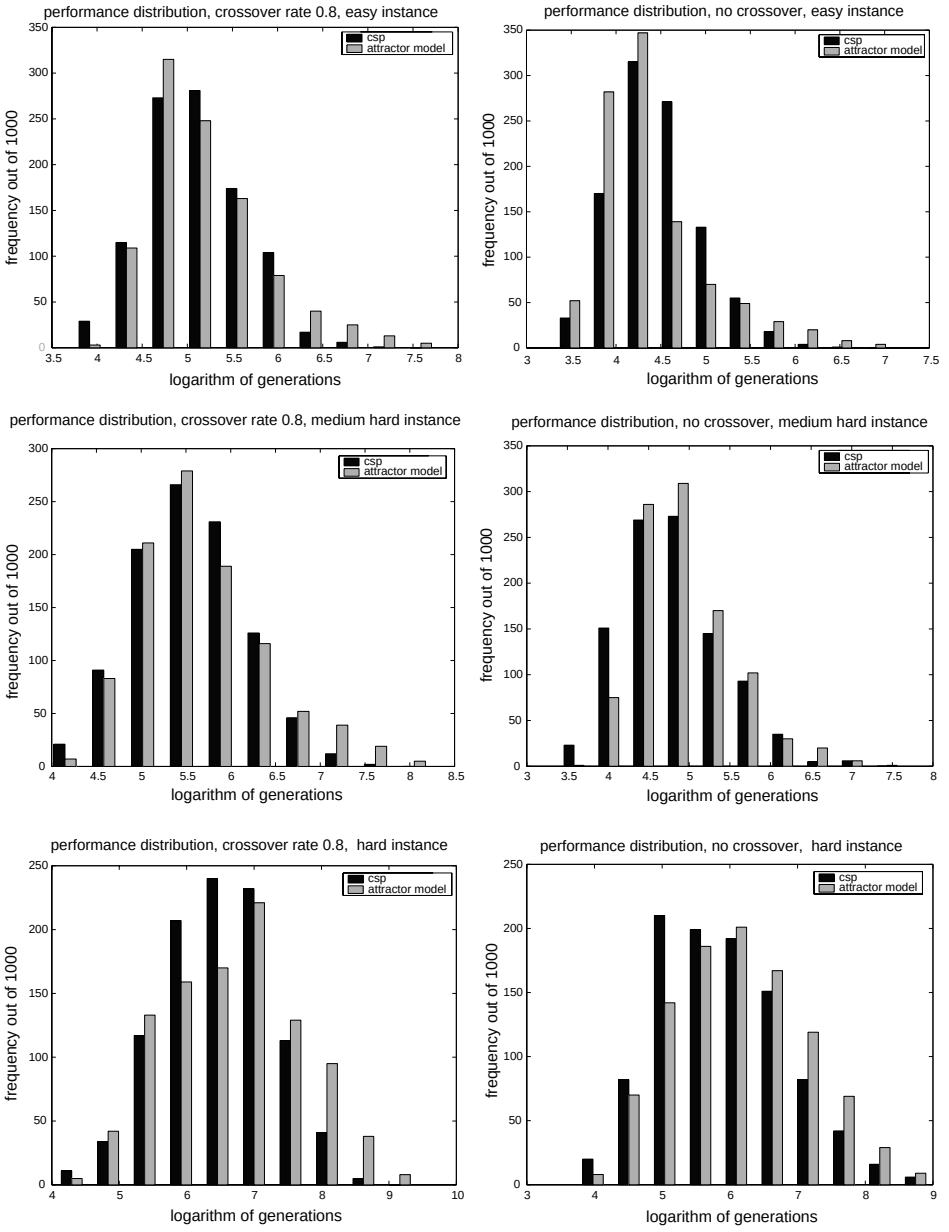

Fig. 6. Performance distributions of the GA on three CSP instances (rows: easy, medium hard, hard) and instances of the attractor model which match for a crossover rate of 0.8 (first column). When crossover is switched off (second column), the distributions remain similar, but the match looses quality. Parameter settings for the model are detailed in Section 5.3 and Table 2.

Table 2
Parameter settings for the matching instances of the attractor model

Category	K	ξ	P_1	P_2
Easy	10	0.85	10	15
Medium hard	10	0	24	28
Hard	48	0	21	22

But the fact that we can reproduce the performance distribution using well-motivated models that share important characteristics with the CSP instances, makes us believe that we have caught two important aspects of the structure of the instances.

6. Conclusions and further work

The summary of the static landscape analysis of 100 randomly generated CSP instances presented in Section 4.3 gives the impression of a smooth search landscape which is only difficult for a GA because of the lack of guidance toward the solutions. The hardness of an instance correlates well, but not perfectly, with the number of local optima in the mutation landscape. The connectivity in the high fitness band, the amount of deception, and the variance on the first-order contribution also correlate to a similar degree with the hardness. toward the solutions. The hardness of an instance correlates well, but not perfectly, with the number of local optima in the mutation landscape. The connectivity in the high fitness band, the amount of deception, and the variance on the first-order contribution also correlate to a similar degree with the hardness.

We conjecture that the ascent towards the high fitness region is slowed down by collateral noise between the building blocks and the presence of attractors which, although weak, nevertheless distract the GA from a direct path to a solution. Using two models which show a similar performance distribution, we have quantified the amounts of noise and the number and importance of the attractors that are needed to cause a similarly slow ascent.

On the to-do side, we would like to find a noise model which is more stable with respect to crossover rate variation, and investigate the stability of both models when other parameters, such as mutation rate or selection pressure, are varied. We have actually created a model for the second phase, based on one primary attractor and several randomly generated secondary attractors at Hamming distance 3 from the primary attractor. This ensures that neutral mutation paths exist between the attractors. Unfortunately, this model is very unstable with respect to crossover rate variation, and we do not believe that it fully reflects the landscape elements that cause the slow search for a solution. This also calls for further research.

The ultimate goal of this line of research is to sketch an accurate picture of the CSP search landscape as the GA sees it, and to construct a classifier that can quantify to which extent search problems match this picture.

Appendix A. The Walsh transform

This appendix briefly recalls the Walsh or discrete Fourier transform.

Let r be an nth primitive root of unity. We start by defining the n^ℓ-dimensional complex matrix $\boldsymbol{V}_{n,\ell}$ containing the Walsh functions for an alphabet of size n and strings of length ℓ. The elements v_{ij} of $\boldsymbol{V}_{n,1}$ are defined as $v_{ij}=r^{ij}$. For example $\boldsymbol{V}_{1,1}=(1)$,

$$\boldsymbol{V}_{2,1}=\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} \quad \text{and} \quad \boldsymbol{V}_{3,1}=\begin{pmatrix}1 & 1 & 1\\ 1 & r & r^2\\ 1 & r^2 & r\end{pmatrix}.$$

Matrix $\boldsymbol{V}_{n,\ell}$ is constructed as $\boldsymbol{V}_{n,\ell}=\boldsymbol{V}_{n,1}^{\otimes\ell}$, with $\boldsymbol{V}^{\otimes\ell}$ denoting the tensor product of ℓ copies of the matrix $\boldsymbol{V}$. The tensor product of two matrices $\boldsymbol{A}=(a_{ij})\in\mathbb{R}^{n\times m}$ and $\boldsymbol{B}=(b_{ij})\in\mathbb{R}^{p\times q}$ is defined as

$$\boldsymbol{A}\otimes\boldsymbol{B}=(a_{ij}\boldsymbol{B})=\begin{pmatrix}a_{00}\boldsymbol{B} & \dots & a_{0(m-1)}\boldsymbol{B}\\ \vdots & \ddots & \vdots\\ a_{(n-1)0}\boldsymbol{B} & \dots & a_{(n-1)(m-1)}\boldsymbol{B}\end{pmatrix}.$$

We denote by $\boldsymbol{I}_{n,\ell}$ the identity matrix of size n^ℓ.

Lemma 1. *For any positive integer ℓ, we have $\bar{\boldsymbol{V}}_{n,\ell}\boldsymbol{V}_{n,\ell}=n^\ell\boldsymbol{I}_{n,\ell}$ where $\bar{\boldsymbol{V}}_{n,\ell}$ denotes the complex conjugate of the matrix $\boldsymbol{V}_{n,\ell}$.*

When $\boldsymbol{f}$ is the vector of function evaluations of a fitness function f, the Walsh transform is defined as $\boldsymbol{w}=\boldsymbol{W}_{n,\ell}\boldsymbol{f}$ with $\boldsymbol{W}_{n,\ell}=n^{-\ell/2}\mathbf{V}_{n,\ell}$. The components of the vector $\boldsymbol{w}$ are called the Walsh coefficients of f. The Walsh functions can be defined as $\psi_t(s)=r^{st}$. The Walsh coefficients permit to recover $\boldsymbol{f}$ because an application of the lemma leads to

$$\boldsymbol{f}=\bar{\boldsymbol{W}}_{n,\ell}(\boldsymbol{W}_{n,\ell}\boldsymbol{f})=\boldsymbol{W}_{n,\ell}\boldsymbol{w}.$$

Appendix B. The Walsh transform of binary CSPs

Suys [18] calculates the Walsh coefficients w_i of functions of the form

$$f(s)=\sum_{0\leqslant i<j<\ell} g_{ij}(s_i,s_j), \quad s\in\Sigma^L.$$

Binary CSPs can easily be written in this form: $g_{ij}(s_i,s_j)$ is set to one if there is a constraint between variables i and j and a conflict between the values s_i and s_j, otherwise it is set to zero.

The Walsh coefficients are indexed according to the formula an^i+bn^j—other coefficients are necessarily zero. When both a and b are zero, the average over the entire

search space is taken and denoted by w_0. It equals

$$w_0 = \frac{1}{n^2} \sum_{\substack{0 \leqslant i < j < \ell \\ a,b \in \Sigma}} g_{ij}(a,b).$$

When a or b are zero, but not both, the index refers to a first-order Walsh coefficient which is computed by

$$\overline{w_{kn^i}} = \frac{1}{n^2} \sum_{a \in \Sigma} r^{ka} \left[\sum_{\substack{0 \leqslant p < i < \ell \\ c \in \Sigma}} g_{pi}(c,a) + \sum_{\substack{i < q < \ell \\ d \in \Sigma}} g_{iq}(a,d) \right].$$

When both a and b are non-zero, we have second-order coefficients

$$\overline{w_{an^i+bn^j}} = \frac{1}{n^2} \sum_{c,d \in \Sigma} r^{ac+bd} g_{ij}(c,d).$$

References

[1] D. Achlioptas, L. Kirousis, E. Kranakis, D. Krizanc, M. Molloy, C. Stamatiou, Random constraint satisfaction: A more accurate picture, Proceedings of the Third Internatnational Conference on Principles and Practice of Constraint Programming, 1997.

[2] T. Bäck, D.B. Fogel, T. Michalewicz, Evolutionary Computation 1: Basic Algorithms and Operators, Institute of Physics Publishing, Bristol and Philadelphia, 2000.

[3] P. Cheeseman, B. Kanefsky, W.M. Taylor, Were the really hard problems are, Proceedings of the IJCAI-91, 1991, pp. 163–169.

[4] B.G.W. Craenen, A.E. Eiben, E. Marchiori, Solving constraint satisfaction problems with heuristic-based evolutionary algorithms, Proceedings of the Congress on Evolutionary Computation, 2000, pp. 1571–1577.

[5] A.E. Eiben, J.I. van Hemert, E. Marchiori, A.G. Steenbeek, Solving binary constraint satisfaction problems using evolutionary algorithms with an adaptive fitness function, in: A.E. Eiben, T. Bäck, M. Schoenauer, H.-P. Schwefel (Eds.), Proceedings of the Fifth Conference on Parallel Problem Solving from Nature, Lecture Notes in Computer Science, Vol. 1498, Springer, Berlin, 1998, pp. 201–210.

[6] D.E. Goldberg, Genetic algorithms and Walsh functions: Part I: a gentle introduction Complex Systems 3 (1989) 129–152.

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

[8] D.E. Goldberg, K. Deb, J.H. Clark, Genetic algorithms, noise and the sizing of populations, Complex Systems 6 (1992) 333–362.

[9] J.J. Grefenstette, Deception considered harmful, in: L.D. Whitley (Ed.), Foundations of Genetic Algorithms 2, Morgan Kaufmann, Los Altos, CA, 1993, pp. 75–92.

[10] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, 1975.

[11] T. Jones, S. Forrest, Fitness distance correlation as a measure of problem difficulty for genetic algorithms, in: L.J. Eshelman (Ed.), Proceedings of the Sixth International Conference on Genetic Algorithms, Morgan Kaufmann, Los Altos, CA, 1995, pp. 184–192.

[12] L. Kallel, B. Naudts, C.R. Reeves, Theoretical aspects of evolutionary computing, Properties of fitness functions and search landscapes, (Chapter) Natural Computing Series, Springer, Berlin, Heidelberg, New York, 2001, pp. 175–206.

[13] S.A. Kauffman, Adaptation on rugged fitness landscapes, Lectures in the Sciences of Complexity, SFI studies, Vol. I, Addison-Wesley, Reading, MA, 1989, pp. 619–712.

[14] E. MacIntyre, P. Prosser, B. Smith, T. Walsh, Random constraint satisfaction: theory meets practice, Proceedings of the Fourth Internattional Conference on Principles and Practice of Constraint Programming, 1998.
[15] B. Manderick, M. de Weger, P. Spiessens, The genetic algorithm and the structure of the fitness landscape, in: R.K. Belew, L.B. Booker (Eds.), Proceedings of the Fourth Internattional Conference on Genetic Algorithms, Morgan Kaufmann, Los Altos, CA, 1991, pp. 143–150.
[16] P. Prosser, Hybrid algorithms for the constraint satisfaction problem, Comput. Intell. 9 (3) (1993) 268–299.
[17] J.L. Shapiro, Statistical mechanics theory of genetic algorithms, in: L. Kallel, B. Naudts, A. Rogers (Eds.), Theoretical Aspects of Evolutionary Computing, Springer, Berlin, 2001, pp. 87–108.
[18] D. Suys, A mathematical approach to epistasis, Ph.D. Thesis, Department of Mathematics and Computer Science, University of Antwerp, Belgium, 1998.
[19] E. Tsang, Foundations of Constraint Satisfaction, Academic Press, New York, 1993.
[20] M.D. Vose, The Simple Genetic Algorithm: Foundations and Theory, Complex Adaptive Systems. Bradford Books, Cambridge, MA and London, England, 1998.
[21] D. Whitley, Fundamental principles of deception in genetic search, in: G.J.E. Rawlins (Ed.), Foundations of Genetic Algorithms, Morgan Kaufmann, Los Altos, CA, 1991, pp. 221–241.
[22] C. Williams, T. Hogg, Exploiting the deep structure of constraint problems, Artif. Intell. 70 (1994) 73–117.