Guided genetic algorithm: a new template concept

Hajer BEN OTHMAN\textsuperscript{a1}, Moncef TAGINA\textsuperscript{b}

\textsuperscript{a,b}COSMOS laboratory, the national school of computer sciences, the university of Manouba, Tunisia

Abstract

Guided genetic algorithm and dynamic distributed double guided genetic algorithm are based on nature laws and by the Neo-Darwinism theory. These evolutionary approaches were very successful addressing Maximal Constraint Satisfaction Problems (Max-CSPs). Our work is inspired by a little mistake when dealing with these two algorithms guidance. In fact these approaches are guided by the min-conflict heuristic and the template concept. The used template is distorted. So, we introduce a new template concept in order to allow a better guidance. We suggest considering the percentages of violated constraints in place of their number. This concept is, then, applied to guide the genetic algorithms. In this paper, we compare the latter guided genetic algorithm with our new template guided genetic algorithm. The experimentations show that our new template guidance improves the optimization process to find best solutions in better time.

© 2015 The Authors. Published by Elsevier B.V.
Peer-review under responsibility of KES International.

Keywords: Maximal constraint satisfaction problems, template concepts, guided genetic algorithm

1. Introduction

CSP formalism consists of variables associated with domains and constraints involving subsets of these variables. A CSP solution is an instantiation of all variables with values from their respective domains. The instantiation must satisfy all constraints. This solution is costly to get and does not necessarily exist within every problem. In such cases, one had better search an instantiation of all variables that satisfies the maximal number of constraints. Such problems called Maximal CSPs and referred to as Max-CSPs, make up the framework to this paper.

Max-CSPs have been dealt with by complete or incomplete methods. The first ones, such as extended forward checking algorithm [4] are able to provide an optimal solution. Unfortunately, the combinatorial explosion thwarts this advantage. The second ones, such as Genetic Algorithms (GAs) [7] have the property to avoid the trap of local optima. They also sacrifice completeness for efficiency. There is other distributed GAs known as Distributed Guided Genetic Algorithm (GGA), Distributed Double Guided Genetic Algorithm (D\textsuperscript{2}G\textsuperscript{2}A) and Dynamic Distributed Double Guided Genetic Algorithm (D\textsuperscript{2}G\textsuperscript{2}A). These approaches have been successfully applied to Max-CSP[4,5]. The latters outperform the centralized Genetic Algorithms (CGAs), which are especially known to be expensive. Our interest in GAs is also motivated by their proven usefulness in many fields [9,12].

This paper aims to enhance the GGA by a new template concept. And it is organized as follows: The next subsection recalls the Guided Genetic Algorithm: the basic concepts and the global dynamic. The following section details both experimental design and results. Finally, concluding remarks and possible extensions to this work are proposed.

1 Hajer BEN OTHMAN. Tel.: +216 71 600 444; fax: +216 71 600 449.
E-mail address: hajerb2@orange.fr
2. Canonical Genetic Algorithms

Genetic algorithms are stochastic search methods that borrow some concepts from nature, they are inspired by the principles of natural Evolution [20,21,22]. CGA maintains a population-pool \( P_i \) of potential solutions called strings, individuals or chromosomes. Each chromosome \( C_p \) is a collection of \( n \) building blocks known as genes, which are instantiated at the beginning with random values from a finite domain or according to some rules. An objective function often referred to as fitness is defined and evaluated for these chromosomes. According to fitness-based heuristics, chromosomes are continuously selected and crossed until a stopping criterion (Fig. 1).

The selection step selects chromosomes from the population-pool to be parents and enters them into the mating-pool according to their fitness by using a technique such as roulette-wheel selection [2], stochastic universal sampling, truncation selection [26] and tournament selection [1].

Two parent chromosomes are randomly selected from the mating-pool to be crossed according a cross-over operator such as single-point cross-over, multi-point-cross-over, uniform cross-over [20,21]. For the single-point cross-over, given a pair of parent chromosomes, an arbitrary cutoff point is picked. Then, the two parents exchange their portions of genes beyond this cutoff point. The cross-over point can assume values in the range 1 to \( L-1 \) (let \( L \) be the chromosome length).

![Fig. 1. The Canonical Genetic Algorithm](image)

The new chromosomes produced undergo a mutation and are also replaced with a mutation probability. An example of mutation consists in randomly choosing a gene in the chromosome and changing its content. In the binary-based code, it consists in flipping a gene from 0 to 1 or from 1 to 0. After the mutation phase, chromosomes are put in the offspring pool. Note that for application of both crossover and mutation probabilities we use a replacement randomly generated value which should be greater than this probabilities experimentation.

Finally, the members of the population pool are compared with those of the offspring pool, this is called the evaluation step. The chromosomes are compared via their fitness value to derive a new population, where the weaker chromosomes may be eliminated and the fittest one is selected as the best. The heuristic for assessing the survival of each chromosome into the new population is called the replacement strategy. In fact, this population corresponds to a new generation. So, CGA progresses through generations until a certain criterion (fixed number of generations, time limit or the solution) is reached.

So, the performance of CGAs tightly depends on initial population, encoding, both cross-over and mutation probabilities, replacement strategy and number of generations. The objective is then to find out the best compromise between quality and complexity.

3. GUIDED GENETIC ALGORITHM

3.1. Basic principles

The relationship between both genetic and CSP formalisms are outlined as follows: each chromosome (respectively gene) is equivalent to CSP potential solution (respectively variable). Moreover, each allele corresponds to a value. On the other hand, each chromosome is attached to a template [29] that is made up of weights referred to as template\(_{i,j}\). Each one of them corresponds to gene\(_{i,j}\) where \( i \) refers to the chromosome and \( j \) to the position. template\(_{i,j}\) represents the number of violated constraints by gene\(_{i,j}\). Templates are in the min-conflict heuristic to guide genetic algorithms [12]. Template concept is successfully used to guide many well known genetic algorithms [7,8,29]. These weights are updated through the penalty operator.
3.2. Our New Template concept versus Tsang’s template concept

Tsang’s template concept is distorted. In fact, if a variable V1 appears in 100 constraints and it has a value V11 which violates 2 constraints. The weight of V1 in the template is equal to 2 ($\delta_i, V1=2$). If simultaneously another variable V2 appears only in 2 violated constraints with its given value V12. The weight $\delta_i, V2=2$. In this case V1 and V2 have a same probability to be replaced. But here we can clearly see that V2 value violates 100% of its involved constraints whereas, V1 value violates only 2% of its involved constraints.

So, one has better use these percentages in place of the old weights so a template$_i,j$ will be equal to the percentages of violated constraints over all the total number of involved constraints. Figure 2 is an example of our template concept used in the same CSP.

![Fig. 2. Our template concept versus Tsang’s template concept](image)

In the CSP example we deal with in figure 2, only constraints C1 and C2 are satisfied by the represented chromosome. Figure 2 It is tempting to note that the belonging to Tsang’s Template concept the optimization process should replace V2, V3 or V5 because they have the greatest values of Template$_i,j$, i.e. these variables violate the greatest number of constraints (2). It is important to note that this choice is done to really intensify the optimization process. But if we use our new templates, it becomes clear that only V3 should be replaced. In fact, the degree of V2, i.e. the number of constraints it appears, is 3. And it violates 2 over 3 constraints. It is the same case for V5. But V3 violates 2 over 2 constraints. Let us remember that the degree of V3 is 2. Here on has better select V3 for the Min-conflict heuristic and not V2 or V5.

3.3. Global Dynamic

In the first step of the guided genetic algorithm we randomly generate the initial population. Then we perform the optimization processes. This starts by initializing all chromosomes corresponding templates. After that we carry out its genetic process. This process, which will be detailed in the following subsection, returns a population “pop” that has been submitted to the crossing and mutating steps only once, i.e. corresponding to one generation. For each chromosome of pop, we compute the number of violated constraints “nvc”. And So on until a stopping criterion is attained or until we meet a chromosome violating zero constraints. At the end of the optimization process we determine the best chromosome namely the one which violates the minimal number of constraints.

3.4. Detailed genetic process

This process differs from the canonical GA described in Fig. 1 in terms of the use of both templates and min-conflict-heuristic [7,8,29]. It starts with determining the mating-pool which consists of pairs of chromosomes, randomly selected by means of the matching procedure (Fig. 4 line 1). Out of each pair of chromosomes, the
crossover operator produces a new child as described in Fig. 5. The child inherits the best genes, i.e. the “lighter” ones, from its parents. The probability, for a parent chromosome, \( (i_1 \text{ or } i_2) \), where \( \text{sum} = \text{template}_{i_1,j} + \text{template}_{i_2,j} \) to propagate its gene\( i,j \) to its child chromosome is equal to 1-\( \text{template}_{i,j}/\text{sum} \). This confirms the fact that the “lighter” genes, i.e. having the smallest number of violated constraints, are more likely than the other to be passed to the child. For each one of its chromosomes selected according to the mutation probability \( P_{\text{mut}} \) (Fig. 8 line 2), \( \text{Species}_n \) uses the min-conflict-heuristic (Fig. 8 line 3) first to determine the gene (variable) involved in the maximal number of violated constraints (Fig. 8 line 1), secondly to select from this gene domain the value that violates the minimal number of constraints (Fig. 7 lines 2-7) and finally to instantiate this gene with this value (Fig. 9 line 8).

It is tempting to note that the change of both gene (variable) and allele (value) is dynamic. Indeed, it depends on other variable values relative to the same chromosome. So, different variables (Fig. 7 line 1) as well as different values may be chosen (Fig. 7 lines 2-7). Thanks to these changes, new values may appear and consequently new chromosomes (that were not in the initial population) may occur. Thereby, the population will be more diversified thus enhancing the search and giving it more chance to converge.

If the obtained chromosome does not violate any constraint (Fig. 8 line 4), we stop the whole process (Fig. 8 line 5). Thus, we display the quoted chromosome. Otherwise, this chromosome is added to the offspring pool and we carry on with the genetic process. Note that the stopping criterion may occur at anytime of the optimization process.

---

**Main optimization process (initial-population)**

1. for \( i := 1 \) to number-of-generations do
2. template-updating (initial-population)
3. pop \( \leftarrow \) genetic-process (initial-population)
4. for each chromosome in pop do
5. nvc \( \leftarrow \) compute-violated-constraints (chromosome)

**Fig. 3. Main optimization process**

**Genetic process**

1. mating-pool \( \leftarrow \) matching (population-pool)
2. template-updating (mating-pool)
3. offspring-pool-crossed \( \leftarrow \) crossing (mating-pool)
4. offspring-pool-mutated \( \leftarrow \) mutating (offspring-pool-crossed)
5. return offspring-pool-mutated

**Fig. 4. Genetic process**

**Crossing (mating-pool)**

1. if (mating-pool size < 2) then return mating-pool
2. for each pair in mating-pool do
3. if (random \([0,1]\) < \( P_{\text{cross}} \) ) then offspring \( \leftarrow \) cross-over (first-pair, second-pair)
4. nvc \( \leftarrow \) compute-violated-constraints (offspring)
5. else offspring-pool \( \leftarrow \) offspring-pool \( \cup \{\text{offspring}\} \)
6. return offspring-pool

**Fig. 5. Crossing process relative to \( \text{Species}_n \)**

**Cross-over (chromosome\(_{i_1}\), chromosome\(_{i_2}\))**

1. for \( j := 1 \) to size (chromosome\(_{i_1}\)) do
2. sum \( \leftarrow \) template\(_{i_1,j} + \) template\(_{i_2,j}\)
3. if (random-integer \([0, \text{sum} - 1]\) \( < \) template\(_{i_1,j}\))
4. then gene\(_{i_3,j} \leftarrow \) gene\(_{i_2,j}\)
5. else gene\(_{i_3,j} \leftarrow \) gene\(_{i_1,j}\)
6. return chromosome\(_{i_3}\)

**Fig. 6. Cross-over operator**
min-conflict-heuristic (chromosomei)
1. \( \delta_{ij} \leftarrow \max (\text{template}_i) \) /* \( \delta_{ij} \) is associated to gene\( i,j \) which is in turn associated to the variable \( v_j \)*/
2. \( \text{nvc}^* \leftarrow \text{nc} \) /* \( \text{nc} \) is the total number of constraints*/
3. for each value in domain of \( v_j \) do
4. \( \text{nvc} \leftarrow \text{compute-violated-constraint} (\text{value}) \)
5. if (\( \text{nvc} < \text{nvc}^* \))
6. then \( \text{nvc}^* \leftarrow \text{nvc} \)
7. value* \( \leftarrow \) value
8. value (gene\( i,j \)) \( \leftarrow \) value*
9. update (template\( i \))
10. return nvc*

Fig. 7. Min-conflict-heuristic relative to chromosomei

mutating (offspring-pool)
1. for each chromosome in offspring-pool do
2. if (random [0,1] < \( P_{\text{mut}} \))
3. then \( \text{nvc}^* \leftarrow \text{min-conflict-heuristic} (\text{chromosome}_i) \)
4. if (\( \text{nvc}^* = 0 \))
5. then stop-process (chromosome)
6. else offspring-pool-mutated \( \leftarrow \) offspring-pool-mutated \( \cup \{\text{chromosome}_i\}\)
7. return offspring-pool-mutated

Fig. 8. Mutating process relative to Species\( \text{n} \)

3.5. Our approach complexity

In the worst case, the crossover’s complexity is equal to \( O(n^2) \). The complexity of the mutation, in the worst case, is also \( O(n^2) \). The complexity of the process in a single generation is equal to the complexity of crossover added to the complexity of mutation so in the worst case it is equal to \( O(n^2) \). For \( N \) generation complexity is equal to \( O(N \times n^2) \).

4. Experimentations

4.1. Experimental Design:

The implementations have been done with ACTALK (Briot, 1989), a concurrent object language implemented above the Object Oriented language SMALLTALK-80. Our work aims to apply our approach for the Max-CSPs. These experimentations are performed on binary CSP-samples randomly generated. The generation is guided by classical CSP parameters: number of variables (\( n \)), domain size (\( d \)), constraint density \( p \) (a number between 0 and 100\% indicating the ratio between the number effective constraints to the number of all possible constraints, i.e., a complete constraint graph) and constraint tightness \( q \) (a number between 0 and 100\% indicating the ratio between the number of forbidden pairs of values (not allowed) by the constraint to the size of the domain cross product). As numerical values, we use \( n = 20 \), \( d = 20 \). Having chosen the following values 0.1, 0.3, 0.5, 0.7, 0.9 for the parameters \( p \) and \( q \), we obtain 25 density-tightness combinations. For each combination, we randomly generate 30 examples. Therefore, we have 750 examples. Moreover and considering the random aspect of genetic algorithms, we have performed 10 experimentations per example and taken the average without considering outliers. For each combination density-tightness, we also take the average of the 30 generated examples. Regarding GA parameters, all implementations use a number of generations (NG) equal to 10, an initial-population size equal to 1000, a cross-over probability equal to 0.5, a mutation probability equal to 0.2 and a random replacement. Note that these parameters values have been proved to be the best because they give the best results. In further works, the sensitivity of experimental result should be experimented versus the fitting of the best GA parameters. In this work the parameter we use are the ones proved to be efficient by other works [9,32].

In this case we compare our approach to the Guided Genetic Algorithm GGA (Tsang 1998). We compute ratios of GGA and NT-GGA performance using the Run time and the satisfaction, as follows:

\[
\text{CPU-ratio} = \frac{\text{GGA-Run-time}}{\text{NT-GGA-Run-time}} \quad : \text{shows the time complexity.}
\]

\[
\text{Satisfaction-ratio} = \frac{\text{NT-GGA Satisfaction}}{\text{GGA Satisfaction}} \quad : \text{recalls the quality}
\]
In order to have a quick and clear comparison of the relative performance of the two approaches. Thus, GGA performance is the numerator when measuring the CPU time ratios, and the denominator when measuring satisfaction ratio. Then, any number greater than 1 indicates superior performance by NT-GGA.

4.2. Experimental results:

From the CPU time point of view illustrated in Fig. 9, we can see NT-GGA furnished results are better than those given by the GGA. Let us mention that this ratio attains its maximum (21) in case of the over-constrained set of examples (i.e. most strongly constrained). The average value of these ratios is about 5.30.

The transition phase, for which tightness is located around 0.5, corresponds to more significant problems. The location and the sharpness of this area depend on the density parameter. The transition covers the passage from under-constrained problems, which are relatively easy to solve, to over-constrained problems that are relatively easy to prove insoluble (Smith, 1994). The satisfaction point of view is shown in Fig. 10. The Satisfaction ratios average is 1.1. So The NT-GGA always accomplishes more satisfaction than GGA.

We have come to these results thanks to the newer guided optimization process which allows more diversification and, at the same time, it allows a well guided intensification. Also, thanks to our new template concept guidance reducing GA temporal complexity. In fact, in one hand this guidance helps GAs in the solution investigation and helps them to escape from local optima. In the second one, the new template concept we propose intensifies the search helping it to attain, rapidly, better fitness function values.

5. Conclusion and perspectives

Within the framework of Max-CSPs, we developed a new template concept. The latter is used to guide genetic algorithms in place of the early template concept introduced by Tsang. Our template concept is basically inspired by the distorted latter guidance of genetic algorithms. In fact our templates are based on percentages of
constraints violence in place of their total number. This measurement helps us to improve the results given by GGA. Not only is bettered the CPU time but also the solution quality.

Further works would essentially consider our new template concept to improve guidance in other metaheuristics. No doubt further refinement of this approach would allow its performance to be improved.

References