An Adjustment Model in a Geometric Constraint Solving Problem

Reyes Pavón
Dept. de Informática
Universidad de Vigo
Campus Universitario As Lagoas s/n
32004 Ourense, Spain
pavon@uvigo.es

Fernando Díaz
Dept. de Informática
Universidad de Valladolid
Plaza Santa Eulalia 9-11
40005 Segovia, Spain
fdiaz@infor.uva.es

M. Victoria Luzón
Dept. de Informática
Universidad de Vigo
Campus Universitario As Lagoas s/n
32004 Ourense, Spain
luzon@uvigo.es

ABSTRACT
An interesting problem related to geometric constraint solving is the choice of the "good" solution. The suitability and effectiveness of genetic algorithms applied to this problem has been demonstrated but their performance depends on the values assigned to their control parameters. Although there are recommendations in the specialisation technical literature about values for these parameters, their optimal settings depend on the problem at hand. Therefore it would be interesting to define a model that automatically adjusts the values of the evolutive parameters as a function of the geometric problem.

This paper proposes a meta-model that generates the recommendations for the right parameter values in genetic algorithms operating as selector mechanism in constructive geometric constraint solvers. It should be stressed that the proposed model is general and automatic. This means that it is applicable to any context and works without the need for any user supervision.

Categories and Subject Descriptors 1.2.6 [Artificial Intelligence]: Learning—parameter learning; I.2.3 [Artificial Intelligence]: Deduction and Theorem Proving—probabilistic reasoning; I.3.5 [Computer graphics]: Computational Geometry and Object Modelling—geometric algorithms, languages, and systems; J.6 [Computer-aided engineering]: Computer-aided design

Keywords Setting parameters, Bayesian Networks, Constructive Geometric Constraint Solving, Genetic Algorithms

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

1. INTRODUCTION

Constraint-based geometric design is one of the paradigms of computer aided design (CAD) that has undergone greatest development in recent years. A geometric constraint problem is defined by a set of points \( P = \{p_1, \ldots, p_{np}\} \) and a finite set of constraints \( C = \{c_1, \ldots, c_{nc}\} \), which are defined over the points. It is to be expected that the set of constraints correctly defines the object, \( nc = 2np - 3 \), and the problem is well-constrained, i.e., it has a finite number of solutions. In this case, solving geometric constraint problems consists of computing the relative position of each point of the geometric object.

Among several techniques for solving geometric constraint problems, the constructive approach is one of the most promising [12]. Constructive solvers decide if a set of simple geometric elements, related by a set of user constraints, consistently defines a geometric object and, if so, whether they determine the position of such geometric elements.

In general, a well-constrained geometric constraint problem potentially has an exponential number of solutions \( S = \{s_1, \ldots, s_{ns}\} \) where \( ns = 2^{op-2} \) [9]. Every instance solution verifies the geometric constraints. However, the user is only interested in one instance that, besides fulfilling the geometric constraints, exhibits some additional properties. This instance solution is called the intended solution. In this context, finding the intended solution (named the Root Identification Problem [12]) means finding an instance solution in the search space [8], that represents the geometric object drawn up by the user.

In [16], Genetic Algorithms (GAs) were shown to be a suitable and effective search mechanism within the root identification problem. The technique over-constrains the geometric problem by defining a set of extra constraints \( R = \{r_1, \ldots, r_{nr}\} \) on the geometric object defined by the user. The extra constraints are used to drive an automatic search of the solution instances space performed by the GA [10]. The search provides a solution instance that maximises the number of extra constraints fulfilled.

Nevertheless, in order to apply a particular GA, a number of components and parameters must be specified by the user: representation of individuals, crossover and mutation operators, and so on. Other data such as the population size \( N \), the crossover rate \( P_c \) and the mutation rate \( P_m \), denoted by control parameters, complete the definition of the GA and are necessary in order to produce an executable version. The values of these parameters greatly determine whether the al-
algorithm will find a "good" solution and whether it will find such a solution efficiently.

Choosing the right parameter values is, however, a hard task. Sometimes parameters are set by experimenting with different values and selecting the ones that give the best results. However, the number of possible parameters and their different values mean that this is a very time-consuming activity. Several references for obtaining the correct control parameters have been reported in GA literature, and it is worth noting that the most suitable choice depends on the application domain [4].

This is the reason why it would be interesting to have a model that, for a given geometric constraint problem, automatically supplies the right values for the control parameters associated with the GA used in the root identification problem, resulting in a practical application for computer aided design.

The paper is focused on defining such a meta-model $M^*$ that recommends the best parameter values $\theta$ (in this case, control parameters values) for any model of interest $M$ (in this case, the GA used in the root identification problem). As shown in the followings sections, Bayesian Networks (BNs) are identified as the suitable formalism to define the meta-model $M^*$, which differs from other systems in that it is:

- general, in the sense that it is a system that can be applied to any model
- automatic, in the sense that it is a system built from a set of data, without any model user supervision

Moreover, the proposed model has the added benefit of removing the need for, or at least reducing the effect of, user decisions about parameter values.

The structure of the paper is as follows. Section 2 is devoted to describing the proposed model for parameter setting. In Section 3, notions about BNs are briefly presented. Section 4 explains the framework to adjust control parameters using BNs. After this, Section 5 describes the experiments carried out for evaluating the proposed model and the results attained. Finally, in Section 6 some conclusions are drawn and future work is proposed and explained.

2. THE PROPOSED META-MODEL

Let $M(\Theta)$ be a model used to solve a problem $P$ and where $\Theta$ is the set of model parameters or characteristics that the user must set up. We are going to distinguish two types of model parameters: external and internal. The first type are parameters that the user must fix to execute the model, while the second type are parameters established and updated in the model learning process.

In this section a meta-model $M^*$, hereinafter referred to as adjustment model, will be proposed to establish the best values of the external parameters of a model $M$. In Figure 1 a simple diagram illustrates its basic notions.

The model parameters, $\Theta$, that the user must set up in order to solve problem $P$, become the domain variables of the adjustment model $M^*$. Furthermore it is necessary to have a set of data $D$ made up of the external parameter values and a quality measurement of executions carried out by model $M(\Theta)$ using them. As such, the data set $D$ serves as input to the adjustment model $M^*$. The proposed model will start to work with an estimate of the parameter values and it will modify them based on its own experience (given by the history of previous runs with model $M$ to solve the problem with a specific configuration of its parameters $\Theta$).

The design of the model proposed should take into account the following features: it should store explicit knowledge (both qualitative and quantitative) about domain variables and should also encode the necessary information for the correct working of the proposed model (model parameters); it should provide a mechanism that permits reasoning under uncertain conditions; lastly, it should supply automatic or semi-automatic methods that establish the model from a database of cases, updating it when new cases are presented.

Bayesian networks constitute a suitable formalism to satisfy these requests which is why they have been used as a base for building the adjustment model. In fact, BNs offer a sound and practical methodology for automatically discovering probabilistic knowledge in databases and for building intuitive and tractable probabilistic models which can be easily used to solve a wide variety of problems. They supply probabilistic reasoning to propagate evidence and have efficient model learning algorithms [11, 13].

Therefore, the meta-model proposed in this paper uses BNs to automatically adjust the parameters $\Theta$ of a model $M$. This BN is constructed from a database, named database of runs, using a BN learning mechanism. The database of runs stores the parameter values $\Theta = (x_1, x_2, ..., x_p)$ used in the executions of the model $M$ to solve the problem at hand as well as making some efficiency measurements ($x_{p+1}, ..., x_n$).

The constructed network nodes stand for the domain variables (in this case the parameters of the model which must be set up) whereas the network arcs represent probabilistic relationships between them. Once the model has been established, the inference mechanism updates probabilities when evidence is entered into the network. This inference mechanism allows us to obtain the best parameter values (the most probable parameters for achieving a "good" result from the model). Later, the suggested configuration ($x'_1, x'_2, ..., x'_n$) can be used for carrying out new runs of the model $M$, highlighting new observations that are incorporated into the database $D$. Thus, this process updates the proposed model while new cases are still being found.

In the following section BNs are briefly introduced.

3. BAYESIAN NETWORKS

BNs are one of the most important frameworks for representing and reasoning with probabilistic models and they have been applied to many real-world problems [14, 17].

![Figure 1: The meta-model $M^*$ for adjusting the parameters of the model $M$](image-url)
Briefly, given a domain with variables of interest \( X = \{X_1, \ldots, X_n\} \), a BN \( B \) is a probabilistic model which encodes the joint distribution of the domain variables (the probability of every possible event as defined by the values of all the variables). The nodes in a BN represent variables of interest and the links represent causal dependencies among the variables. The dependencies are quantified by conditional probabilities for each node according to its parents in the network. The network supports the computation of the probabilities of any subset of variables, given evidence about any other subset.

The number of possible configurations of the variables of interest may grow exponentially, but BNs achieve compactness by factoring the joint distribution into local, conditional distributions for each variable given its parents. Therefore, the global semantics of BNs specify that the full joint distribution is given by the product

\[
P(x) = P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | pa_i)
\]

where \( pa_i \) denotes a set of values for \( X_i \)’s parents.

There are also entirely equivalent local semantics, which assert that given its parents each variable is independent of its nondescendants in the network. The collection of independent assertions suffices to derive the global assertion in Equation 1, and vice versa. The global semantics lead directly to a variety of algorithms for reasoning. Mainly, from the product specification in Equation 1, one can express the probability of any desired proposition in terms of the conditional probabilities specified in the network. In certain literature many algorithms exist, which make probabilistic inference possible [15].

Local semantics are most useful in constructing BNs, because selecting the direct causes of a given variable as parents automatically satisfies the conditions for local conditional independence. In this sense, the availability of knowledge from domain experts is very important. Since causal relationships among variables are hidden or unknown in many situations, several learning methods have been developed to extract this knowledge from data.

In the proposed adjusting model the inference mechanism is used to give the user the recommendable (most probable) values of the external parameters which the user must set up in order to solve the problem of interest with the model \( M \).

The major feature of the proposed adjusting model \( M^* \) is its ability to adapt its behaviour as soon as new examples are presented. This ability is supported by the learning methods available for BNs. Learning in BNs involves two main aspects: learning probabilities and learning the network structure. The conditional probabilities \( P(x_i | pa_i) \) can be updated continuously from observational data using different methods [11, 5]. It is also possible to learn the structure of the network, using methods that trade-off network complexity against the degree to which the data fits [6, 7, 11].

At present, there are a lot of software tools for manipulating BNs which support structural and parametric learning. Using these tools it is possible to build a meta-model \( M^* \) with parameter setting capacity in a model \( M(\Theta) \) from a database \( D \) of executions for model \( M(\Theta) \), as illustrated in the next section.

4. ADJUSTMENT MODEL DESIGN AND WORKING

This section is focused on illustrating the usefulness of the proposed adjustment model for achieving the right parameter values in GAs that operate as a selector mechanism in constructive geometric constraint solvers. The same study case was broached by Barreiro in [3]. In his work, he completed the work of [16] presenting an extended statistical analysis and his results are used to evaluate how good this adjustment model is.

The issue of setting the control parameter values \( \Theta = \{N, P_m, P_c\} \) of a GA is crucial for good performance. Therefore, these control parameters constitute the domain variables of the proposed adjustment model \( M^* \) for this case study. Moreover, in order to get a complete database \( D \) of previous experiences that serves as input to the proposed model \( M^* \) a battery of runs of the GA was carried out by Barreiro [3], recording in each execution the control parameter values and a quality measure, \( esp - eval \), of such a GA configuration: \( < n, p_m, p_c, esp - eval > \).

Figure 2 shows the proposed adjustment model applied to one concrete geometric problem and a specific GA. In the following subsections some details about the modules identified in this figure are presented.

4.1 The genetic algorithm

With regard to the particular GA used in the root identification problem, the first stage was to decide on a representation of a candidate solution to the problem. Since each solution instance generated by the solver could be represented by an index \( I = \{i_1, \ldots, i_j, \ldots, i_{np}\} \) where each component \( i_j \in \{-1, 1\} \), binary representation was the best choice.

The next step was to determine the fitness function, which was defined as the number of extra constraints that satisfy an instance solution for the solution space generated by the solver.

Stochastic universal sampling [2] was the selection scheme used in conjunction with elitism, in an attempt to always maintain the fittest member in the population. The variation operators proposed were the one-point crossover operator and the bit-flipping mutation operator [4].

To obtain a full specification the decision has been made to fill the initial population randomly and terminate the
4.2 The database of runs

The proposed adjustment model needs a database which collects previous runs of the GA. This database serves as input to the learning process. This is why, for each geometric problem studied, a DB has been built from the executions and data collected by Barreiro in his analysis. In his work, due to the stochastic nature of GAs, Barreiro carried out 150 observations for each combination of parameters \( < p_m, p_c, n > \). With the rank of values used for the variables of interest (see Table 1), the number of possible configurations of the control parameters is 504 (7x8x9 levels). Therefore, each geometric problem creates a database with 75600 (150x504) cases.

For each one of these cases the GA was run 200 times, with a different seed every time, calculating in each execution the number of evaluations necessary to find a ”good” solution to the root identification problem, working out the next equation:

\[
\text{esp - eval} = \hat{E}(\text{eval}) = \frac{1}{k} \sum_{i=1}^{k} \text{eval}_i + \frac{n-k}{k \times \text{eval}_{\text{max}}} \tag{2}
\]

where \( n \) stands for the number of runs, \( k \) is the number of successful runs, \( \text{eval}_{\text{max}} \) represents the termination condition and \( \text{eval}_i \) means the number of evaluations in the \( i^{th} \) run. So, a function that measures the level of quality of each GA configuration is obtained.

This quality value of each case was also stored in the database \( < p_m, p_c, n, \text{esp - eval} > \). This domain variable was previously discretized using a logarithmic approach. In this way, the database was ready to start the learning process.

4.3 The learning and inference process

With respect to the learning phase of the adjustment model, BNs are a good candidate because they can reveal information about problem parameters and their interaction. Moreover, since the selected problem has a limited size and the number of parameters is reduced, the cost of such learning is low.

Among the different standard learning methods that exist, the K2 algorithm has been chosen to carry out the experiments [7]. With respect to the inference mechanism, the selected propagation method was Hugin [13].

Now that the data and algorithms used in the experimentation are known, the working of the adjustment model, applied to the root identification problem is illustrated in Figure 2. The database \( D \) stores the values for the control parameters and the expectation of the evaluations of previous runs of the GA used to find the intended solution of the geometric problem at hand. From this database and prior knowledge, a BN is constructed using the K2 learning algorithm [7]. The network nodes stand for the control parameters of GA and the quality measure of such configuration, whereas the network arcs represent probabilistic relationships among these domain variables. Once the model is established, the inference mechanism supplies the most probable configuration for the control parameters \( < N, P_c, P_m > \) in order to minimize the number of evaluations carried out by the GA to find the intended solution. The fewer the evaluations necessary, the more efficient the GA. Later, these control parameters could be used for carrying out new runs of the GA, bringing about further observations that are incorporated into the database \( D \). Thus, the learning process successively updates the BN as new cases are gathered.

5. METHODOLOGY AND RESULTS

Initially, the experiments carried out to test the adjustment model described previously start from the nine geometric problems used in the Barreiro’s work, characterized by a number of points \((np)\) equal to 20 and extra geometric constraints \((nr)\) ranked among 26 and 50. For each geometric problem a database of runs was constructed with the 75600 observations compiled in [3] and later preprocessed. Once the data was ready, the K2 algorithm was used. The structure of the learnt BNs from the available databases of runs was the same for the nine cases and is shown in Figure 3. The network structure corroborates that the three control parameters \( (P_m, P_c, N) \) have a significant influence on the expectation of the evaluations \( \hat{E}(\text{eval}) \), as Barreiro also identifies in his study.

![Figure 3: BN inferred from the available database of runs (75600 cases)](image)

Nevertheless, with the aim of applying the GA to the root identification problem in a more efficient way, it is necessary to ascertain which control parameter values are the most suitable for the problem at hand. In this study case, the most suitable values are those ones which minimize the number of evaluations, and in this sense, the Hugin propagation method is applied.

As a result of this inference process, the combination of control parameters proposed by our adjustment model for each examined geometric problem is shown in Table 2, in the column entitled \( \text{BAM} \) - Bayesian Adjustment Model. Besides this, and with the aim of comparing the results with the ones obtained by Barreiro’s statistical analysis, his optimal control parameters are also shown in Table 2 and are
referred to as STAT. A third column, named DB, contains the parameter configuration of the database which presents the minimum expectation of evaluations on average. This method acts as the reference model or golden standard for the comparison of the other models.

In order to compare the three suggested configurations for each geometric problem, several statistical analyses have been made and results of these are shown in Table 3.

Firstly, for each analyzed geometric problem, a normality test was applied to the observations selected. The test indicated that the observations were not normally distributed. Then, the Kruskal-Wallis non-parametric test was applied to detect differences among configurations. This test revealed that in four geometric problems (problems n2, n3, n5 and n6) no differences existed, whereas in the remaining problems there were very significant differences between the observations, which are numerically shown from column 2 to 5 in Table 3.

Once the figures with significant differences among the proposed configurations were identified, a Mann-Whitney non-parametric test was performed for all combinations of pairs of samples in order to detect homogeneous groups. The results of this test are also displayed in Table 3, from column 6 to 8, where $x$ indicates significant differences and * indicates significant but not very significant differences.

So, the optimal configuration of parameters proposed by the adjustment model is the same as the configuration proposed by the database algorithm. This result can be corroborated by Table 2 where, in these geometric problems, both algorithms suggest the same parameter values.

Thus, in the geometric problems n1, n4, and n9 the DB and BAM algorithms constitute a homogeneous group, both algorithms suggest the same parameter values. In the geometric problem n7, the difference between the configuration proposed by the adjustment model and the statistical configuration is significant but not very significant ($p = 0.0902$). In contrast, no similarities were found among control parameter values in the geometric problem n8.

Therefore, the proposed adjustment model obtains results very similar to the STAT method with regards to the DB method, which is considered the golden standard. Moreover, it should be emphasized that the proposed adjustment model is automatic and no user supervision is required.

Another interesting issue, not breached by Barreiro, concerns the adaptation of the model, namely how the optimal configuration of parameters could change when more experience is gained. In order to address how the adjustment model changes when more cases are incorporated into the database of runs, we have looked at a situation where a database increases progressively. Initially, the number of cases in the database of runs was reduced to 504 (the different configurations that exist). While the adjustment model is at work, new cases are obtained at the same time, changing the context in which it is used. In all, four different situations were analyzed, where the number of cases incorporated into the database were, successively, 504, 504x25, 504x50 and 504x74.

With this test, the adaptation of the proposed model is evaluated and the BNs obtained are shown in Figure 4.

In this case, the BN automatically adapts its structure and parameters to each particular database of runs. Thus, the recommendations concerning the best GA parameter configurations are different in each stage of the adjustment model built. The more cases the database stores, the more accurate the parameter values become.

The results obtained with this problem allow us to draw some conclusions with regard to parameter setting in general.

### 6. CONCLUSIONS AND FUTURE WORK

In this paper, a meta-model for parameter setting based on BN has been presented. The adjustment model has been designed with the aim of reducing the effort of the user who needs to optimise the parameters $\Theta$ of the model $M(\Theta)$ which solves the problem $P$. The main novelty of this proposed model resides in the fact that it is general, i.e. domain independent, and automatic, that is to say, no user supervision is necessary. Another advantage is that the proposed model is easy to implement.

It was also observed that basing the adjustment model on BNs may be an appropriate solution. Learning a BN from data allows the acquisition of knowledge implicit in the domain variables and successive updating of this knowledge as new cases are collected. Also, the inference mechanism over BN allows recommendations about the best parameter values under uncertain conditions to be given.

Our experiments showed that this proposed model could be successfully applied to GAs that operate as a selector mechanism in the root identification problem. The results obtained were similar to the ones provided by Barreiro’s study and the ones extracted after an analysis of the database. Moreover, it is adaptive, so that the more data collected, the more precise the results become.

As for future work, and taking into account that right parameter values of the model $M(\Theta)$ may change according to the problem at hand, it would be interesting if the described adjustment model allowed these values $\Theta$ to be obtained from characteristics of the problem $P$. It might be an effective way of extending the utility of this proposed model. To put this idea into practice, an adequate solution would be to combine the adjustment model with a database of several instances of the same problem, in the way that each case deals

<table>
<thead>
<tr>
<th>Geometric figure</th>
<th>BAM configuration</th>
<th>STAT configuration</th>
<th>DB configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case n1</td>
<td>(0.9, 0.5, 40)</td>
<td>(0.9, 0.5, 100)</td>
<td>(0.9, 0.5, 40)</td>
</tr>
<tr>
<td>Case n2</td>
<td>(0.9, 0.3, 30)</td>
<td>(0.9, 0.5, 40)</td>
<td>(0.9, 0.3, 40)</td>
</tr>
<tr>
<td>Case n3</td>
<td>(0.9, 0.2, 30)</td>
<td>(0.9, 0.2, 30)</td>
<td>(0.9, 0.2, 40)</td>
</tr>
<tr>
<td>Case n4</td>
<td>(0.9, 0.7, 20)</td>
<td>(0.9, 0.5, 70)</td>
<td>(0.9, 0.7, 20)</td>
</tr>
<tr>
<td>Case n5</td>
<td>(0.9, 0.3, 40)</td>
<td>(0.9, 0.6, 60)</td>
<td>(0.9, 0.2, 60)</td>
</tr>
<tr>
<td>Case n6</td>
<td>(0.9, 0.3, 40)</td>
<td>(0.9, 0.3, 60)</td>
<td>(0.9, 0.2, 60)</td>
</tr>
<tr>
<td>Case n7</td>
<td>(0.9, 0.8, 40)</td>
<td>(0.9, 0.8, 50)</td>
<td>(0.9, 0.3, 40)</td>
</tr>
<tr>
<td>Case n8</td>
<td>(0.9, 0.6, 20)</td>
<td>(0.9, 0.2, 20)</td>
<td>(0.9, 0.4, 20)</td>
</tr>
<tr>
<td>Case n9</td>
<td>(0.9, 0.5, 50)</td>
<td>(0.9, 0.5, 70)</td>
<td>(0.9, 0.5, 50)</td>
</tr>
</tbody>
</table>

Figure 4: BNs inferred: a) 504 cases; b) 13104 cases; c) 38304 cases; d) 75600 cases
with a specific BN. In this sense, the CBR (Case Based Reasoning) methodology is considered the most suitable mechanism for guiding that integration [1] allowing users to extrapolate and adapt the parameter values learned for other types of similar problems. The CBR systems have produced good results in other domains to solve similar problems and they are considered an appropriate methodology for setting the parameters of a model, provided that it has different instances of the particular problem to solve. In this sense, the current paper sets the foundations to extend the work to the ideal situation, where learning the best parameters for one particular problem instance could tell us a great deal about the best parameters for another instance of the same problem.

7. ACKNOWLEDGMENTS

Victoria Luzón has been partially supported by the Ministerio de Educación y Ciencia and by FEDER under grant TIN2004-06326-C03-01.

8. REFERENCES