Statistical modelling of CSP solving algorithms performance[⋆]

Student: Carles Mateu
Supervisors: Ramon Béjar and Cèsar Fernández

Dpt. d'Informàtica, Universitat de Lleida, Jaume II, 69, E-25001 Lleida, Spain,
{ramon,cesar,carlesm}@eup.udl.es

Abstract. The goal of this work is to try to create a statistical model, based only on easily computable parameters from the CSP problem to predict runtime behaviour of the solving algorithms, and let us choose the best algorithm to solve the problem. Although it seems that the obvious choice should be MAC, experimental results obtained so far show, that with big numbers of variables, other algorithms perfom much better, specially for hard problems in the transition phase.

1 Introduction

Our goal is to characterize and to be able to predict the search cost, of some of the most important CSP algorithms and heuristics when solving CSP problems by obtaining a statistical model of the algorithm runtime based on inexpensively computed parameters obtained from the CSP problem specification and the associated constraints and nogoods graphs.

Such a model will give us three important items concerning the studied CSP problems. First, the model provides a tool to predict the search cost of a given instance, allowing a portfolio of solvers to decide for the best algorithm before to proceed. Second, the models will give an insight about which are the main features that characterize the complexity of a RBCSP. Finally, another potential benefit of the model is pointing out which features are the algorithms most sensible to, thus helping to guess potential areas of improvement.

This work follows a close related methodology used for SAT problems, described in [1, 2]. In a first step, we define a broad benchmark scenario that covers a full range of cases of random binary CSP (RBSCP) problems. We proceed by solving a large set of instances of each problem, using 3 different algorithms and 3 different variable ordering heuristics for each one. This first analysis gives already a initial insight about what type of algorithms performs better according to the size of the problem. Then we define a set of features to be analyzed in conjunction with the time performance, some directly related to the problem specification parameters and some others related to the structure of the

[⋆] Research partially supported by projects TIC2003-00950 and TIN2004-07933-C03-03 funded by the *Ministerio de Educación y Ciencia.*

constraints and nogoods graphs. Such a combination of time performance and feature measurements is then analyzed in order to obtain a statistical model based on regression analysis.

2 Experimental design

2.1 Problems

Random Binary Constraint Satisfaction Problems We use model B problems , defined by the following parameters ($\langle n, d, c, t\rangle$), where n is the number of variables, d the domain cardinality, c the number of pairs of constrained variables and t the tightness of each constraint. We have generated problems in 4 different regions according to their density ($c = \alpha$ and $c = 3\alpha$) and domain cardinality ($d = n/5$ and $d = n/2$), being $\alpha = \frac{n \log(d)}{\log(2)}$. Such a minimum value of c determines that model is not flawed asymptotically as n increases At each region, we have generated easy and hard problems (varying n), so leading to 8 families of problems. For each family, we have generated problems above and below the phase transition region, located approximately at $t_{crit} = d^2\left(1 - d^{-n/c}\right)$, giving the following distribution:

	$c = \alpha$	$c = 3\alpha$
$d = n/5$	$\langle 70, 14, 273, 72 : 116\rangle$	$\langle 50, 10, 500, 5 : 40\rangle$
	$\langle 60, 12, 215, 52 : 92\rangle$	$\langle 40, 8, 360, 1 : 30\rangle$
$d = n/2$	$\langle 40, 20, 173, 149 : 249\rangle$	$\langle 30, 15, 352, 26 : 66\rangle$
	$\langle 30, 15, 117, 72 : 152\rangle$	$\langle 20, 10, 199, 1 : 40\rangle$

Table 1. Families of generated problems

Finally, we have generated 1.000 instances per problem, which leads to a total of 415.000 instances to solve and analyze.

2.2 Algorithms

All the instances mentioned above have been solved using 3 different algorithms; forward checking (FC) , forward checking with conflict directed backjumping (FC_CBJ) and Mantaining Arc Consistency (MAC) , each one of them using three different variable ordering heuristic; `dom`, `dom/deg` and `dom+deg` . Those 9 computational procedures spent more that 24 weeks of CPU time for the completion of the generated instances.

We have found that `dom/deg` heuristic outperforms the other two heuristics in all the problems checked as pointed in [3], meanwhile there are substantial differences between run-times of algorithms depending on the parameters of the problem thus justifying the need for a criteria for choosing between those algorithms.

Problem size parameters	
(1 – 8)	Number of variables (n), domain size (d), constraints (c),
	and nogoods per constraint (t), and their corresponding inverse values
(9–20)	Ratios $n/d, n/c, n/t, d/c, d/t, c/t$ and their corresponding inverse values
(21–22)	Value $\|t - t_{crit}\|$ and its inverse
Constraint and nogood graph structure parameters	
(23–26)	constraint and nogood graph width and their inverse values
(27–42)	constraint $((27-34))$ and nogood graph $((35-42))$
	vertex degree statistics (mean, stdev, max, min) and their inverse values
(43–58)	constraint $((43-50))$ and nogood graph $((51-58))$
	vertex clustering coefficient statistics and their inverse values
(59–62)	constraint $((59-60))$ and nogood graph $((61-62))$ upper bounds
	for their cycle cutset number and their inverse values
(63–64)	constraint graph upper bound for its treewidth and their inverse value
(65–66)	constraint graph upper bound for its fill number and their inverse value
Constraint structure parameters	
(67,71,75,	fraction of constraints that are invariant under the max, min,
79,83,87)	discriminator, dual discriminator, switching, and median functions
(68,72,76,	and their corresponding inverse values.
80,84,88)	
(69,73,77,	*degree of invariability* of the constraints to the same previous 6 functions
81,85,89)	
(70,74,78,	and their corresponding inverse values.
82,86,90)	

Table 2. CSP instance parameters used to find the predictive models

3 Parameters of the model

We use three different kinds of parameters to try to capture the structure of the CSP: problem size parameters, graph structure parameters and constraint structure parameters. Table 2 shows all the parameters used, labelled with the numbers we use to refer to them in the tables that show the best predictive models we have found. For all the parameters we use both its value and the inverse value in the models.

In the first set of parameters, we include the basic CSP instance features; number of variables (n), domain size (d), number of constraints (c) and number of nogoods per constraint (t). We also include different ratios between these values: $n/d, n/c, n/t, d/c, d/t, c/t$ in our models. And finally, we also include the value $|t - t_{crit}|$. In the second set of parameters, we include a series of graph parameters that have been shown to be relevant to the worst-case complexity of some CSP algorithms, when considering the constraint graph. However, we also obtain the same parameters for the nogood graph [1]. For the vertices of the

[1] We define the nogood graph as the graph with one vertex for every possible assignment for a single variable and an edge between any pair of vertices such that they define a nogood of the CSP instance.

graph, we calculate the node degree statistics (mean, standard deviation, min and max). We also consider, for each vertex, its clustering coefficient, a parameter originally used in [4] to characterize small-world networks. Two very important structural parameters for dynamic CSP algorithms are the treewidth [5] and the cycle cutset number [6]. Because is NP-hard to obtain these values, we have used heuristic approximation algorithms that provide an upper bound for them. For the treewidth we have used the *Lexicographic Breatdth First Search, variant Minimal (LEX_M)* algorithm proposed in [7] that also approximates a related parameter, the minimum fill-in of the graph [2]. At the same time, we compute the width of the graph, using the algorithm proposed in [6], that is always a lower bound for the treewidth.

The third kind of parameters included in the model are algebraic properties based on the notion of polymorphism of a relation, that is a homomorphism of Cartesian powers of a relation to the relation itself. When a function is a polymorphism of a relation, we also say that the relation is invariant under the function or that the relation is closed under the function. So, because almost all the advanced CSP algorithms used today achieve some level of consistency, it turns out that the polymorphisms of a CSP can be a key factor for explaining the performance of the algorithms.

4 Statistical analysis

Once the time performance and the parameter measurements have been obtained, we define a methodology analysis to obtain statistical models able to explain performance behavior. As in [2], we have used only linear regression analysis because of its computational efficiency, but including the following modifications that extend the ability of the model. First, as shown in section 3, we include in the model the inverse of each feature and the ratios of the four model B parameters. Second, we perform a cubic expansion for each feature, such that

$$y \sim \sum_i x_i + \sum_i x_i^2 + \sum_i x^3 + \sum_{i,j,k} x_i : x_j : x_k + \sum_{i,j} x_i^2 : x_j .$$

And third, we check linear and logarithmic models, achieving logarithmic ones better fits than linear for all the checked problems. Finally, we use the forward selection method to obtain the most important set of features.

Table 3 shows the 6 most important features obtained from a logarithmic regression test. In the last row, we show the coefficient of determination (R^2). The range is $0 \leq R^2 \leq 1$, being values closer to 1 better fits. Those values of the coefficient of determination show that good prediction can be obtained by using a few parameters. Observe that for all the algorithms the values obtained for R^2 are very similar, thus indicating that the models are of comparable quality.

[2] We thank the authors of [7] for providing us their implementation of the treewidth algorithm

	FC			FC_CBJ			MAC		
	`dom`	`dom+deg`	`dom/deg`	`dom`	`dom+deg`	`dom/deg`	`dom`	`dom+deg`	`dom/deg`
1st.	(21)	(21)	(21)	(21)	(21)	(21)	(21)	(21)	(21)
2nd.	(39)	(25)	(35)	(39)	(39)	(39)	(39)	(39)	(35)
3rd.	(63)	(64)	(65)	(63)	(63)	(63)	(64)	(63)	(63)
4th.	(28)	(28)	(89)	(28)	(28)	(28)	(17)	(17)	(90)
5th.	(19)	(90)	(2)	(90)	(90)	(19)	(10)	(3)	(2)
6th.	(22)	(22)	(90)	(89)	(89)	(22)	(35)	(22)	(22)
R^2	0.847	0.849	0.873	0.870	0.873	0.874	0.868	0.867	0.910

Table 3. Most important features for the logarithmic model using forward selection variable method and the corresponding coefficient of determination

Looking at the parameters that best predict the behaviour of the algorithms we observe some interesting coincidences and differences. For all the algorithms, the most important parameter is always the $|t - t_{crit}|$ value, thus showing that the position of the problem with respect to the phase transition is a very informative value. The second most important parameter is the minimum degree of the nogood graph or a directly related or correlated parameter[3]. The next considered parameter is upper bound of the tree width of the constraint graph (or its inverse), except in one case where its a correlated parameter [4]. This parameter has been studied previously in the context of dynamic CSP algorithms that perform tree-like decompositions of the CSP [8]. For the rest of parameters, the discrepancies increase, showing that the rest of parameters cannot give a uniform explanation for the behaviour of all the algorithms. Observe that in the models the nogood graph seems to capture very relevant information, this shows the importance of increasing our understanding of structural properties of the CSP nogood graph (traditionally, CSP community has focused its research in the constraint graph).

As mentioned above, using 6 parameters in our regression models give us good fits. But which is the best we can do? Figure 1 helps to answer this question. There, the evolution of the coefficient of regression is plotted as the subset of features in our regression test is increased, for MAC_`dom/deg`. The plots for the other algorithms are very similar.

5 Conclusions

So far we have been able to create a model that has a reasonable quality for predicting runtime behaviour of the algorithms analysed. The model created for RBCSP problems includes, as the most significant parameters, the position of the problem respect the phase transition point, the nogood graph minimum

[3] The width of the graph depends on the degrees of the vertices of the graph.

[4] The min-fill upper bound, computed using the same algorithm than the tree width.

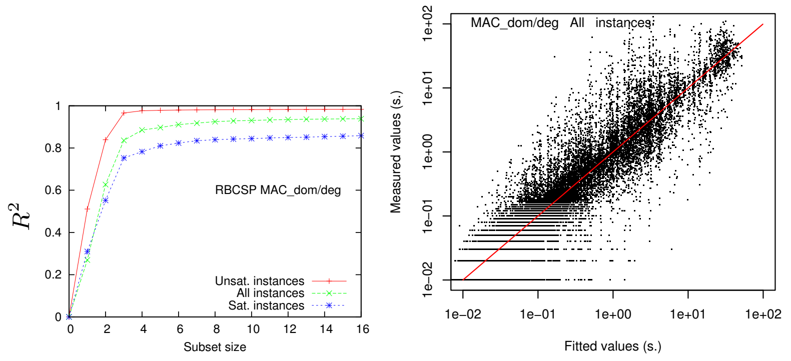

Fig. 1. (a) Evolution of the coefficient of regression (R^2) for RBCSP problems with MAC_`dom/deg` when using all, satisfiable or unsatisfiable instances in the model (b) Measured vs. predicted runtimes for instances using MAC_`dom/deg` algorithm

degree and an upper bound of the constraint graph tree width (or some related parameter).

Further research is being carried to increase the quality of the model, adding more relevant parameters, especially graph related ones, so the model can be used with a higher degree of confidence. We are also extending this work to model other kinds of more structured problems beyond RBCSP, namely QWH.

References

1. Leyton-Brown, K., Nudelman, E., Shoham, Y.: Learning the empirical hardness of optimization problems. In: Proceedings CP'02, Ithaca NY (2002) 556–572
2. Nudelman, E., Leyton-Brown, K., Hoos, H., Devkar, A., Shoham, Y.: Understanding random SAT: beyond the clauses-to-variable ratio. In: Proceedings CP'04, Toronto, Canada (2004) 438–452
3. Bessière, C., Régin, J.: MAC and combined heuristics: two reasons to forsake FC (and CBJ?) on hard problems. In: Proceedings CP'96, Cambridge MA (1996) 61–75
4. Watts, D.J., Strogatz, S.H.: Collective Dynamics of small-world networks. Nature **393** (1998) 440–442
5. Bodlaender, H.L., Gilbert, J.R., Hafsteinsson, H., Kloks, T.: Approximating treewidth, pathwidth, frontsize, and shortest elimination tree. Journal of Algorithms **18-2** (1995) 238–255
6. Dechter, R.: Constraint Processing. Morgan Kaufmann (2003)
7. Koster, A.M., Bodlaender, H.L., van Hoesel, S.P.: Treewidth: Computational Experiments. ZIB-Report **01-38** (2001)
8. Koster, A.M.C.A., van Hoesel, S.P.M., Kolen, A.W.J.: Solving partial constraint satisfaction problems with tree decomposition. Networks **40** (2002) 170–180