Optimal Refutations for Constraint Satisfaction Problems

Tudor Hulubei and Barry O'Sullivan
Cork Constraint Computation Centre
Department of Computer Science, University College Cork, Ireland
tudor@hulubei.net, b.osullivan@cs.ucc.ie

Abstract

Variable ordering heuristics have long been an important component of constraint satisfaction search algorithms. In this paper we study the behaviour of standard variable ordering heuristics when searching an insoluble (sub)problem. We employ the notion of an optimal refutation of an insoluble (sub)problem and describe an algorithm for obtaining it. We propose a novel approach to empirically looking at problem hardness and typical-case complexity by comparing optimal refutations with those generated by standard search heuristics. It is clear from our analysis that the standard variable orderings used to solve CSPs behave very differently on real-world problems than on random problems of comparable size. Our work introduces a potentially useful tool for analysing the causes of the heavy-tailed phenomenon observed in the runtime distributions of backtrack search procedures.

1 Introduction

The conventional wisdom in systematic backtrack search is that a variable ordering heuristic should be chosen so that if a bad assignment is made, and search enters an insoluble (sub)tree, the heuristic will prefer variables that have a high likelihood of proving quickly that the current (sub)tree is indeed unproductive. This property of a variable ordering heuristic is often referred to as *fail-firstness* [Haralick and Elliott, 1980]. Fail-firstness can be seen as an explanation for why particular variable ordering heuristics are better than others for solving CSPs, and it has received considerable attention from the research community [Smith and Grant, 1998; Beck *et al.*, 2004]. Various ways of explaining the quality of variable ordering heuristics in terms of fail-firstness have been proposed. Some authors argue that heuristics that minimise branching depth are best [Haralick and Elliott, 1980], while others argue that such a policy can cause an increase in branching factor which can have the effect of increasing the size of the resultant (sub)tree [Smith and Grant, 1998].

A search algorithm enters an insoluble search tree either because the problem itself is insoluble, or because the current partial solution has just been extended with an assignment that cannot lead to a complete solution. In either case, the ideal is that insolubility can be proven with the least amount of effort. In this paper we will measure such effort by the number of nodes visited to prove that an insoluble (sub)tree has no solution. We can evaluate variable ordering heuristics by comparing the number of nodes that they visit to prove the insolubility of a (sub)problem with the minimum number of nodes required to draw the same conclusion.

By comparing optimal refutations with those generated by MAC [Sabin and Freuder, 1994], when combined with various standard variable ordering heuristics, we show empirically that there is a significant difference between the performance of heuristics on typical random problems and those found in the real world. We also suggest that small-to-medium sized random binary CSPs are of limited value when studying and developing variable ordering heuristics. While this has been the intuition amongst constraints researchers, our analysis shows empirically that this is indeed the case and gains insights into the nature of the difference in behaviour between random and real-world problems. Finally, we discuss how our results may provide further insights into the causes of the heavy-tailed phenomenon observed in the runtime distributions of backtrack search procedures. For example, by focusing on the distributions of actual and optimal refutations we can study whether heavy-tailedness is present because of the variable ordering heuristic, the value ordering heuristic, or is more fundamentally related to the structure of the problem.

2 Preliminaries

Definition 1 (Constraint Satisfaction Problem). *We define a* CSP *as a 3-tuple* $P \doteq \langle V, D, C \rangle$ *where* V *is a finite set of* n *variables* $V \doteq \{V_1, \ldots, V_n\}$, D *is a set of finite domains* $D \doteq \{D(V_1), \ldots, D(V_n)\}$ *such that* $D(V_i)$ *is the finite set of possible values for* V_i, *and* C *is a finite set of constraints such that each* $C_{ij} \in C$ *is a subset of* $D(V_i) \times D(V_j)$ *specifying the combinations of values allowed between* V_i *and* V_j, *where* $i < j$. *We say that* P *is arc-consistent (AC) if* $\forall v_k \in D(V_i)$ *and* $\forall j$ *such that* $C_{ij} \in C$, $\exists v_l \in D(V_j)$ *with* $(v_k, v_l) \in C_{ij}$. *An* assignment $A_{ik} \doteq \langle V_i = v_k \rangle$ *represents a reduction of* $D(V_i)$ *to* $\{v_k\} \subseteq D(V_i)$. *A* solution *to* P *is a set of distinct assignments* $\mathcal{S} \doteq \{A_{l_1 k_1}, \ldots, A_{l_n k_n} | (v_{k_i}, v_{k_j}) \in C_{ij}\}$.

Definition 2 (Search Algorithm). *A search algorithm* $\Theta \doteq \langle \Lambda, \Delta, \prec_V, \prec_v \rangle$ *is a combination of a branching method* Λ, *a*

consistency enforcement method Δ, a variable ordering $\prec_V$ and a value ordering $\prec_v$, both of which can be either static or dynamic.

Definition 3 (Search Tree). *A search tree $\mathcal{T}$ for a problem P is a set of nodes and arcs. Each node corresponds to a set of assignments, $\mathcal{N} \hat{=} \{A_{l_1k_1}, \ldots, A_{l_{p-1}k_{p-1}}, A_{l_pk_p}\}$, totally ordered by a dynamic variable ordering heuristic $\prec_V$. The root of the search tree is a special node $\mathcal{R} \hat{=} \emptyset$. Two nodes $\mathcal{N}_1$ and $\mathcal{N}_2$ are connected by an arc if $\exists A_{ij}$ such that $\mathcal{N}_2 = \mathcal{N}_1 \cup A_{ij}$, in which case we say that $\mathcal{N}_1$ is the parent of $\mathcal{N}_2$, and $\mathcal{N}_2$ is the child of $\mathcal{N}_1$. For every node $\mathcal{N}$, its children are totally ordered by a dynamic value ordering heuristic $\prec_v$.*

Search trees are defined in the context of a specific search algorithm. For a particular CSP instance P, and search algorithm Θ, a one-to-one mapping exists between the nodes in the search tree $\mathcal{T}$ and the assignments made by Θ.

Definition 4 (Mistake Point). *For a soluble problem P, a mistake point $\mathcal{M}$ is a node identified by a set of assignments $\mathcal{M} \hat{=} \{A_{l_1k_1}, ..., A_{l_{p-1}k_{p-1}}, A_{l_pk_p}\}$, totally ordered by $\prec_V$, for which $\exists S \in \mathcal{S}$ such that $\mathcal{M} \setminus \{A_{l_pk_p}\} \subset S$, but $\neg\exists S \in \mathcal{S}$ such that $\mathcal{M} \subset \mathcal{S}$. Since an insoluble problem does not admit any solutions, we define the mistake point associated with an insoluble problem as the root of its search tree.*

Informally, a mistake point corresponds to an assignment that, given past assignments, cannot lead to a solution even though, in the case of a soluble problem, a solution exists. Whenever the value ordering heuristic makes such a mistake, the role of the variable ordering heuristic is to guide the search out of that insoluble search tree as quickly as possible. However, it is important to realise that the actual set of mistake points encountered during search is also dependent on the variable ordering heuristic used.

For a soluble problem P, let $P_{\mathcal{M}} \hat{=} \{V_{\mathcal{M}}, D_{\mathcal{M}}, C_{\mathcal{M}}\}$ be the insoluble (sub)problem corresponding to $\mathcal{M}$, where $V_{\mathcal{M}} \hat{=} V \setminus \{V_{l_1}, ..., V_{l_p}\}$, $C_{\mathcal{M}} \hat{=} \{C_{ij} | V_i, V_j \in V_{\mathcal{M}}, C_{ij} \in C\}$, and $D_{\mathcal{M}}$ is the set of current domains after arc-consistency has been restored to reflect the domain reductions due to $\mathcal{M}$. If P is insoluble, as a notational convenience, we define $\mathcal{M} \hat{=} \emptyset$ and $P_{\mathcal{M}}$ as the arc-consistent version of P. For brevity, we will refer to the *insoluble (sub)tree* rooted at a mistake point and its corresponding *insoluble (sub)problem* interchangeably.

Definition 5 (Refutations). *Given a search algorithm Θ, a **refutation** for a given insoluble (sub)problem $P_{\mathcal{M}}$, rooted at mistake point, $\mathcal{M}$, is simply the corresponding search tree $\mathcal{T}_{\mathcal{M}}$. We will refer to $|\mathcal{T}_{\mathcal{M}}|$, the number of nodes in $\mathcal{T}_{\mathcal{M}}$, as the* size *of the refutation.*

We study the refutations found using a version of MAC that uses AC-3 [Mackworth, 1977] for consistency enforcement and selects values randomly. Our goal is to determine how close to optimality are the refutations obtained when well known variable ordering heuristics, with randomly broken ties, are substituted for $\prec_V$. For each heuristic $\prec_V$ we first collect the mistake points it generates when using MAC (note that each variable ordering heuristic will generate a different set of mistake points). When we analyse MAC in conjunction with a certain $\prec_V$ on a mistake point $\mathcal{M}$, we will refer to the refutation for the (sub)problem $P_{\mathcal{M}}$ as the *actual refutation*. We will contrast the actual refutation with the *optimal refutation* for $P_{\mathcal{M}}$, obtained by replacing $\prec_V$ with a new variable ordering heuristic $\overline{\prec_V}$ such that $|\mathcal{T}_{\mathcal{M}}|$ is minimised.

Sometimes, rather than using the optimal refutation, we compare against the *quasi-optimal*, defined as the smallest refutation whose depth does not exceed that of the actual. By selecting variables based on a depth-first traversal of the tree of minimum size, $\overline{\prec_V}$ causes MAC to generate the smallest possible search tree proving insolubility for $P_{\mathcal{M}}$. In our experience it is extremely rare that the quasi-optimal refutation is larger than the optimal. By accepting quasi-optimality we can use the depth of the actual refutation as an upper bound, dramatically speeding-up search for better refutations.

Definition 6 (Improvement Ratios). *For a given mistake point, the **refutation improvement ratio** is the ratio between the size of the actual refutation and that of the optimal refutation. The **overall refutation improvement ratio** is simply the ratio between the total number of nodes in all the actual refutations and the total number of nodes in all their corresponding optimal refutations encountered when finding the first solution or proving insolubility.*

3 Analysing Refutations

Our focus here is only on how a variable ordering heuristic behaves when trying to *prove insolubility* for a (sub)problem. We are not interested in focusing on *how often* a variable ordering heuristic enters an insoluble (sub)tree. Therefore, our analysis is centered around finding optimal refutations for all the mistake points in a search tree. Unfortunately, this is a herculean task, as we have to search through all the possible permutations of assignments, i.e. consider all possible search trees rooted at each mistake point. Without clever optimisations, this would not be feasible for anything but the tiniest problems.

3.1 Basic Algorithm

Given a problem instance, a search algorithm (MAC in our case) and the variable ordering heuristic to be evaluated, we begin by searching for a solution, with the goal of identifying all the mistake points. The mistake points represent the roots of the insoluble (sub)trees the search algorithm happens to encounter before finding the first solution. For each such mistake point we compute both the actual refutation, by rerunning MAC on the current (sub)problem using the heuristic we are evaluating, and the optimal refutation, by considering all the possible permutations of assignments to the variables involved in that (sub)problem. Note that insoluble instances are just a special case where the only mistake point is before the first assignment. Furthermore, when analysing insoluble problems, it is important to realise that the optimal refutation is independent of the heuristic used; in other words *no other ordering* could possibly prove insolubility more quickly.

Both the search algorithm with the variable ordering heuristics being analysed and the exhaustive search for the

optimal refutation use the same method of enforcing consistency (AC in our case). Failure to do so would make the refutation improvement ratio a function of both the variable ordering and the level of consistency enforced, thus being detrimental to our goal of analysing variable ordering heuristics.

The version of MAC that we use employs k-way branching [Smith and Sturdy, 2004], rather than binary branching, so that selecting a variable V_i creates $|D(V_i)|$ branches in the search tree. Thus, we make sure that refutation sizes are not influenced by the value ordering.

3.2 Optimisations

There are a number of optimisations that we have applied.

Firstly, we can easily maintain an upper bound on the size of the optimal refutation by initially setting it to the size of the actual refutation, then updating it as smaller refutations are found. Therefore, the search for the optimal refutation can be implemented as a branch-and-bound procedure.

Secondly, we can improve the search by looking ahead to avoid choices that cannot improve on the current smallest refutation. Whenever a variable V_i is selected at a certain depth all the values in its domain have to be tried, and they all have to fail for the current (sub)problem to be proved insoluble. Consequently, we know that by selecting V_i, the size of the current partial refutation will increase by at least a number of nodes equal to $|D(V_i)|$. We call this a 1-level look-ahead.

By temporarily assigning to V_i, in turn, every value v in its domain, and by attempting to restore arc-consistency after every such assignment, we can associate with each v a minimum contribution to the size of the refutation. If the assignment makes the subproblem arc-inconsistent, v's contribution will be 1, given by the node corresponding to the assignment itself. However, if arc-consistency can be restored after the assignment, at least one more variable will have to be considered before the current subproblem can be proved insoluble. Therefore, v will carry a minimum contribution equal to the smallest domain size amongst all the remaining unassigned variables. We call this a 2-level look-ahead.

In general, V_i's selection would increase the size of the current partial refutation by at least the sum of the minimum contributions of all the values in its domain. If that would cause the current partial refutation to exceed the size of the smallest refutation found so far, V_i will be skipped at the current depth. This dramatically reduces the search space and brings reasonable size problems within reach.

4 Experiments

The basic experiment we run is this: for each variable ordering heuristic we first find all the mistake points that MAC visits when finding the first solution or proving insolubility; for each mistake point, we find MAC's actual refutation (still with the same heuristic); using the size of the actual refutation as an upper bound, we find the (quasi-)optimal refutation. This approach was adopted because different variable orderings generate potentially different sets of mistake points.

We have performed experiments[1] on random binary problems, various instances of the n-queens problem for varying values of n, and a subset of the RLFAP Celar Scen11 [Cabon *et al.*, 1999]. We have studied the following dynamic variable ordering heuristics: min-domain [Haralick and Elliott, 1980], max-degree, min(domain/degree) [Bessière and Regin, 1996], and brelaz [Brélaz, 1979]; their corresponding anti-heuristics; and random variable selection. These are regarded as the standard general-purpose variable ordering heuristics used in the CSP literature. Ties were broken randomly. Throughout our experiments we have used a dynamic random value ordering.

Note that for heuristics we report the optimal refutations for random problems and n-queens, and the *quasi*-optimal refutations for RLFAP. For anti-heuristics we always compute *quasi*-optimal refutations. While the true optimal refutations for RLFAP may be smaller than those reported, based on our observations the difference is not significant. Limiting the search to quasi-optimal proofs allowed us to gather significantly more data for this problem.

4.1 Random and n-Queens Problems

Due to the extreme computational requirements of the search for optimal refutations, we had to limit our experiments to random problems with 15 variables and uniform domain sizes of 10 values. We generated 800 instances for each tightness point using a random problem generator that conforms to the Model-B specification [Gent *et al.*, 2001].

We selected instances at the highest peak of difficulty for these problems, found to pass through a plane corresponding to density 0.95. Tightness was varied along this plane. Specifically, we identify a distinct tightness at which we move from generating a set of problem instances that are all soluble to generating instances that are all insoluble. In Figure 1 this transition occurs at a tightness of 0.3. We indicate the range of tightness for which all problems were soluble (resp. insoluble) with a "high" (resp. "low") line.

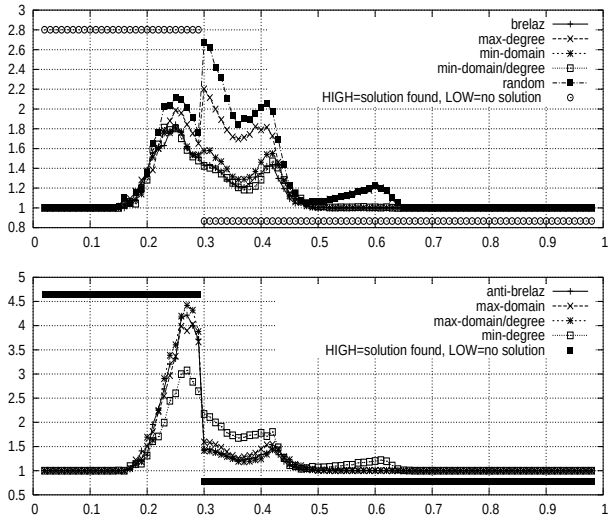

Fig. 1: Mean overall improvement ratios (y-axis) for random problems as a function of tightness (x-axis). Heuristics (anti-heuristics) are in the top (bottom) plot. Median plots are virtually identical.

[1] Code freely available with source at http://hulubei.net/tudor/csp.

In Figure 1 we present the results associated with each of the variable ordering heuristics studied. Note that we are not ranking heuristics, but are simply studying how close their actual refutation sizes are to optimality. The overall refutation improvement ratio peaks at 1.8 for min(domain/degree) at a tightness of about 0.24, meaning that for the set of mistake points encountered when solving these problems using min(domain/degree) there does not exist a variable ordering that could decrease the number of nodes in the corresponding refutations by more than that factor, on the average. Since at the peak of difficulty proving insolubility accounts for most of the time spent in search, this result suggests that no improvement in the ordering of variables, for the problems we have studied, would ever lead to an order of magnitude speedup. Indeed, in the soluble region we can see that at best we can hope to halve the time taken by max-degree. In theory, however, for soluble problems heuristics could do better by failing less.

Once we move through the phase-transition to insoluble problems, all heuristics offer improvement ratios less than 2.2 for max-degree and less than 1.6 for all other heuristics, except random. In the case of min(domain/degree) we see that when using MAC at the peak of difficulty the ratio is approximately 1.4. This is an interesting result since is demonstrates that for proving insolubility *no variable ordering heuristic exists*, for these problems, that can improve upon min(domain/degree) by more than a factor of 1.4 at the peak of difficulty.

Figure 2 presents a plot of the distribution of actual refutation sizes against optimal, per mistake point, for the random and min(domain/degree) variable ordering heuristics. Of interest here is the apparent *linear relationship* between actual versus optimal refutation size. A similar linear relationship was observed for all other heuristics and anti-heuristics studied. When we discuss the performance of variable ordering heuristics on a real-world problem in the next section we will see that this linear relationship does not hold and that for very large refutations found by a heuristic, the optimal refutation can be many orders of magnitude smaller.

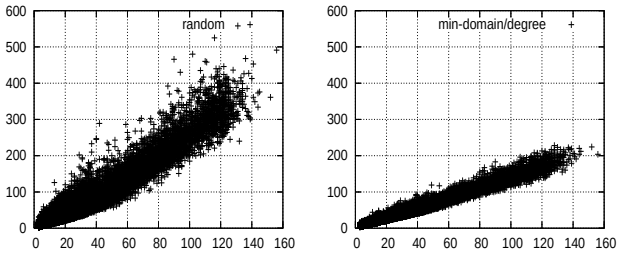

Fig. 2: Refutation sizes for mistake points in random problems when using random and min(domain/degree) variable orderings: actual (y-axis) as a function of optimal (x-axis).

The performance of a random variable ordering is of comparable quality to that of a good variable ordering heuristic in this problem class. From Figure 1 we see that the improvement ratio of the random heuristic is approximately 2.7 at the phase transition, close to that of standard heuristics.

Finally, in Figure 3 we present, for each variable ordering heuristic, the distribution of insoluble (sub)trees in the data-set as a function of the improvement ratio. We see that almost all ratios were close to 1, meaning that almost all of the time the standard variable ordering heuristics performed very close to perfection. We also see that for the random ordering heuristic ratios of 10 are found less that 0.01% of the time and that there does not exist a single insoluble (sub)tree that provides an opportunity to make improvements of more than an order of magnitude over current heuristics.

In experiments performed over a large data-set of insoluble subtrees generated from problems ranging from 8 to 22 queens we have observed similarly small *refutation improvement ratios* for n-queens, only slightly bigger than those observed for random problems. Also, the linear relationship between actual and optimal refutation size was observed, as well as a similar distribution of ratios.

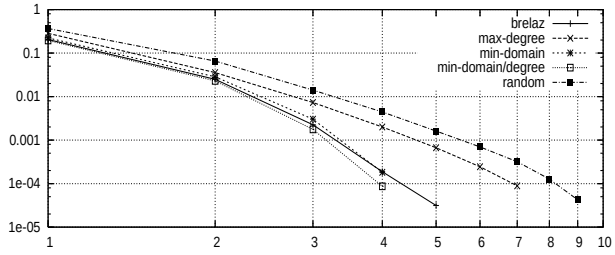

Fig. 3: Probability of an insoluble (sub)tree in the data-set for random problems (y-axis) having an improvement ratio greater than a certain value (x-axis).

4.2 RLFAP Celar Scen11

The small improvement ratios obtained for random problems and n-queens, and estimated in [Li and Gérard, 2000] for hard random unsatisfiable 3-SAT, do not seem to hold for real-world problems. A popular benchmark for binary CSPs, RLFAP Celar Scen11 was an obvious candidate for our analysis. Unfortunately, at 680 variables, we could not analyse the entire problem. Instead, we extracted a subset of the original problem made up of 22 variables that were part of the backdoor set [Williams *et al.*, 2003] over which MAC backtracked while solving the original problem. We analysed all the mistake points found when solving this subset of RLFAP over 500 times for all variable ordering heuristics.

Refutation improvement ratios of 20 and over were observed, even when using the best heuristics, as seen in Figures 4 and 5, while they were non-existent for random and n-queens problems. Comparing Figures 2 and 4 we can clearly see that heuristics behaves very differently on this problem than on the random problems presented earlier. Specifically, we can see that actual refutations correlate very poorly with quasi-optimal and that no linear relationship exists; note that a log-scale had to be used on the y-axis of the left plot in Figure 4 due to the spread of actual refutation sizes for each quasi-optimal point on the x-axis.

However, of particular interest in this problem is the pattern we observe in Figure 5. While the vast majority of refutations

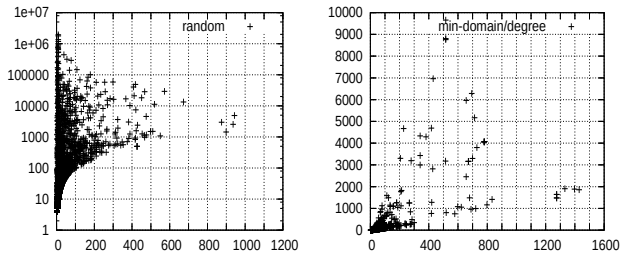

Fig. 4: Refutation sizes for mistake points in the RLFAP data-set when using random and min(domain/degree) variable orderings: actual (y-axis) as a function of quasi-optimal (x-axis).

offer small improvement ratios, it is possible to find subtrees on which the standard heuristics perform very poorly. For example, we can see that in a small number of subtrees max-degree did 100 times worse than quasi-optimal, while the other heuristics, except min(domain/degree), sometimes did 50 times worse than quasi-optimal. Clearly, the behaviour of standard variable orderings is significantly different on real-world problems than on the random problems that many researchers use as a basis for studying these heuristics. In the case of a random variable ordering, we observed a very small number of subtrees where the refutation found was more than a factor of 10^5 worse than quasi-optimal, and that in the vast majority of cases it performed more than an order of magnitude worse than quasi-optimal.

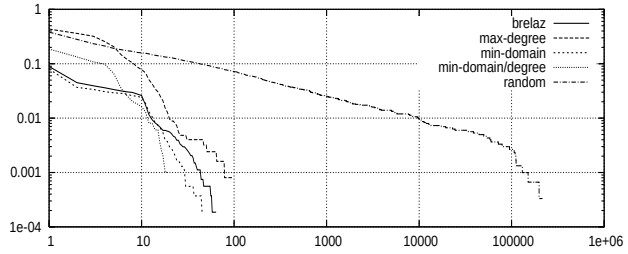

Fig. 5: Probability of an insoluble subtree in the RLFAP data-set (y-axis) having an improvement ratio greater than a certain value (x-axis).

Therefore, our standard heuristics behave very differently, indeed much worse, on real-world problems than on random and n-queens problems. In particular, they can be less robust in the sense that they can sometime waste significant amounts of time finding refutations for insoluble (sub)trees that are far worse than optimal.

5 Discussion

It is clear from our analysis that the standard variable orderings used to solve CSPs behave very differently on real-world problems than on random problems of comparable size, which are frequently used to study the performance of heuristics in the research literature. Our results demonstrate that there are considerable opportunities to improve search performance on real-world problems. Some researchers have already begun developing the necessary infrastructure needed to study real-world problems in a systematic way [Gomes *et al.*, 2000; Walsh, 1999; Walsh, 2001]. There have also been some recent successful results demonstrating that one can benefit considerably from developing variable ordering heuristics that can exploit characteristics one encounters in real-world problems [Boussemart *et al.*, 2004; Dubois and Dequen, 2001; Refalo, 2004].

It is not known whether the linear relationship between actual and optimal refutations is preserved as the size of random problems increases. Recent work on studying the performance of variable ordering heuristics on very large CSPs has shown that there can be orders of magnitude differences between heuristics [Bessière *et al.*, 2001]. However, those results do not contradict the results presented in this paper with respect to the relative performance of standard variable orderings on random and non-random problems.

Our analysis is related to the work of Li and Gerard on studying the limits of branching heuristics for random 3-SAT [Li and Gérard, 2000]. However, our interests here are in studying the limits of variable ordering heuristics in a constraint satisfaction context. Furthermore, our analysis is capable of finding exact optimal refutations, rather than approximating them, and focuses on the differences observed between random and real-world problems. Our work is also closely related to the theoretical analysis of proof complexity for satisfiability and constraint satisfaction [Beame *et al.*, 1998; Mitchell, 2003].

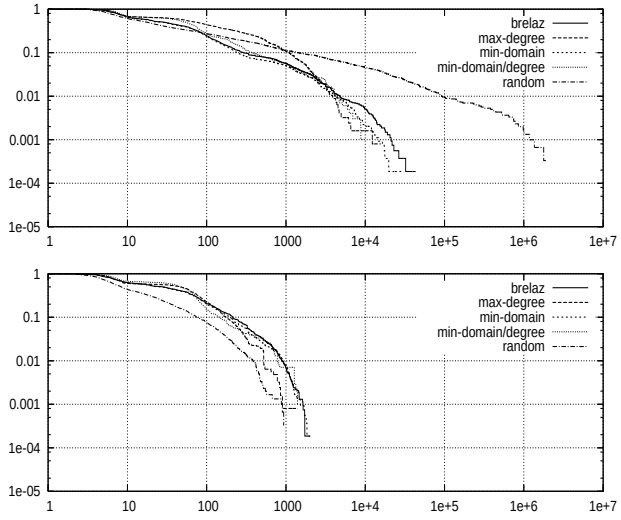

Fig. 6: Probability of an insoluble subtree in the RLFAP data-set (y-axis) being greater than an optimal/actual (top/bottom) refutation of a given size (x-axis).

In this paper we introduce a potentially useful tool for analysing the causes of the heavy-tailed phenomenon observed in the runtime distributions of backtrack search procedures [Gomes *et al.*, 2000]. Figures 5 and 6 show, for RLFAP, a very small percentage of insoluble (sub)problems for which standard variable ordering heuristics will find refutations that are significantly larger than optimal, leading to a

long runtime. Furthermore, it is also interesting to note from Figure 5 that the vast majority of actual refutations are very close to being optimal, with the exception of those found with a random variable ordering heuristic.

Gomes et al. [2004] report a high correlation between the distribution of the depth of mistake points and the runtime distribution, e.g. the presence of heavy-tails. We can further extend this analysis by also considering the size of optimal refutations rooted at these mistake points. This may provide a way of isolating the causes of heavy-tailed behaviour by making a distinction between the effects of the algorithm and its components (propagation method, variable and value orderings), versus the effects of the inherent structure of the problem [Walsh, 1999; Walsh, 2001]. As part of our future work in this area we will study optimal refutations for problems that are well known to exhibit heavy-tail behaviour.

6 Conclusions

In this paper we have proposed a novel approach to empirically looking at problem hardness and typical-case complexity by comparing optimal refutations with those generated by standard search heuristics. We have compared refutations of insoluble (sub)trees found using well known variable ordering heuristics combined with a standard CSP search algorithm against optimal refutations. Our results show that there is a significant difference between the performance of heuristics on typical random problems and those found in the real-world. We suggest that small-to-medium sized random binary CSPs are of limited value to constraints researchers and introduced a potentially useful tool for analysing the causes of the heavy-tailed phenomenon observed in the runtime distributions of backtrack search procedures.

Acknowledgments

This material is based on work supported by Science Foundation Ireland under Grant 00/PI.1/C075. We would like to thank Barbara M. Smith, Eugene C. Freuder, Nic Wilson, Chris Beck, Bart Selman, Carla Gomes and Susan Epstein for their comments and suggestions, and to John Morrison and the Boole Centre For Research in Informatics for providing access to their Beowulf cluster.

References

[Beame *et al.*, 1998] P. Beame, R. Karp, T. Pitassi, and M. Saks. On the complexity of unsatisfiability of random k-cnf formulas. In *Proceedings of ACM STOC-1998*, pages 561–571, 1998.

[Beck *et al.*, 2004] J.C. Beck, P. Prosser, and R.J. Wallace. Trying again to fail-first. In *Proceedings of CSCLP-2004*, LNCS 3419, pages 41–55, 2004.

[Bessière and Regin, 1996] C. Bessière and J-C. Regin. MAC and combined heuristics: two reasons to forsake FC (and CBJ?) on hard problems. In *Proceedings of CP-1996*, LNCS 1118, pages 61–75, 1996.

[Bessière *et al.*, 2001] C. Bessière, A. Chmeiss, and L. Sais. Neighbourhood-based variable ordering heuristics for the constraint satisfaction problem (long version). Available from: http://www.lirmm.fr/~bessiere/pubs , 2001.

[Boussemart *et al.*, 2004] F. Boussemart, F. Hemery, C. Lecoutre, and L. Sais. Boosting systematic search by weighting constraints. In *Proceedings of ECAI-2004*, pages 146–150, 2004.

[Brélaz, 1979] D. Brélaz. New methods to color the vertices of a graph. *Communications of the ACM*, 22(4):251–256, 1979.

[Cabon *et al.*, 1999] B. Cabon, S. Givry, L. Lobjois, T. Schiex, and J.P. Warners. Radio link frequency assignment. *Constraints*, 4(1):79–89, 1999.

[Dubois and Dequen, 2001] O. Dubois and G. Dequen. A backbone-search heuristic for efficient solving of hard 3-SAT formulae. In *Proceedings of IJCAI-2001*, pages 248–253, 2001.

[Gent *et al.*, 2001] I.P. Gent, E. MacIntyre, P. Prosser, B.M. Smith, and T. Walsh. Random constraint satisfaction: Flaws and structure. *Constraints*, 6(4):345–372, 2001.

[Gomes *et al.*, 2000] C.P. Gomes, B. Selman, N. Crato, and H. Kautz. Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. *Automated Reasoning*, 24(1/2):67–100, 2000.

[Gomes *et al.*, 2004] C.P. Gomes, C. Fernández, B. Selman, and C. Bessière. Statistical regimes across constrainedness regions. In *Proceedings of CP-2004*, LNCS 3258, pages 32–46, 2004.

[Haralick and Elliott, 1980] R.M. Haralick and G.L. Elliott. Increasing tree search efficiency for constraint satisfaction problems. *Artifi cial Intelligence*, 14(3):263–313, 1980.

[Li and Gérard, 2000] C.M. Li and S. Gérard. On the limit of branching rules for hard random unsatisfiable 3-sat. In *Proceedings of ECAI-2000*, pages 98–102, 2000.

[Mackworth, 1977] A.K. Mackworth. Consistency in networks of relations. *Artifi cial Intelligence*, 8(1):99–118, 1977.

[Mitchell, 2003] D.G. Mitchell. Resolution and constraint satisfaction. In *Proceedings of CP-2003*, LNCS 2833, pages 555–569, 2003.

[Refalo, 2004] P. Refalo. Impact-based search strategies for constraint programming. In *Proceedings of CP-2004*, LNCS 3258, pages 557–571, 2004.

[Sabin and Freuder, 1994] D. Sabin and E.C. Freuder. Contradicting conventional wisdom in constraint satisfaction. In *Proceedings of ECAI-1994*, pages 125–129, 1994.

[Smith and Grant, 1998] B.M. Smith and S.A. Grant. Trying harder to fail first. In *Proceedings of ECAI-1998*, pages 249–253, 1998.

[Smith and Sturdy, 2004] B.M. Smith and P. Sturdy. An empirical investigation of value ordering for finding all solutions. In *Workshop on Modelling and Solving Problems with Constraints*, 2004.

[Walsh, 1999] T. Walsh. Search in a small world. In *Proceedings of IJCAI-1999*, pages 1172–1177, 1999.

[Walsh, 2001] T. Walsh. Search on high degree graphs. In *Proceedings of IJCAI-2001*, pages 266–271, 2001.

[Williams *et al.*, 2003] R. Williams, C.P. Gomes, and B. Selman. Backdoors to typical case complexity. In *Proceedings of IJCAI-2003*, pages 1173–1178, 2003.