

# Search Heuristics and Heavy-Tailed Behaviour

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.** The heavy-tailed phenomenon that characterises the runtime distributions of backtrack search procedures has received considerable attention over the past few years. Some have conjectured that heavy-tailed behaviour is largely due to the characteristics of the algorithm used. Others have conjectured that problem structure is a significant contributor. In this paper we attempt to explore the former hypothesis, namely we study how variable and value ordering heuristics impact the heavy-tailedness of runtime distributions of backtrack search procedures. We demonstrate that heavy-tailed behaviour can be eliminated from particular classes of random problems by carefully selecting the search heuristics, even when using chronological backtrack search. We also show that combinations of good search heuristics can eliminate heavy tails from Quasigroups with Holes of order 10, and give some insights into why this is the case. These results motivate a more detailed analysis of the effects that variable and value ordering can have on heavy-tailedness. We show how combinations of variable and value ordering heuristics can result in a runtime distribution being *inherently heavy-tailed*. Specifically, we show that even if we were to use an Oracle to refute insoluble subtrees optimally, for some combinations of heuristics we would still observe heavy-tailed behaviour. Finally, we study the distributions of refutation sizes found using different combinations of heuristics and gain some further insights into what characteristics tend to give rise to heavy-tailed behaviour.

## 1 Introduction

The Italian-born Swiss economist Vilfredo Pareto first introduced the theory of non-standard probability distributions in 1897 in the context of income distribution. These distributions have been used to model real-world phenomena, from weather forecasting to stock market analysis. More recently, they have been used to model the cost of combinatorial search methods. Exceptionally hard instances have been observed amongst certain classes of constraint satisfaction problems, such as graph colouring [12], SAT [7], random problems [2, 8, 20, 21], and quasigroup completion problems [10]. In studying this phenomenon, researchers have used a wide range of systematic search algorithms such as chronological backtracking, forward-checking, Davis-Putnam and MAC. It is widely believed that the more sophisticated the search algorithm, the less likely it is that the exceptionally hard problem instances will be observed.

Instances that are exceptionally hard occur in the under-constrained area and are often harder than those in the critically constrained region [21]. For a proper understanding of search behaviour one must study the runtime distributions [8] associated

with either repeatedly solving a single instance with a randomised algorithm, or with solving a large ensemble of instances of some class of problems. Some runtime distributions exhibit an extremely large variance which can be described by heavy-tailed distributions whose tails have power-law decay. Gomes et al. [9] provided an overview of the heavy-tailed behaviour previously observed near the phase transition in NP-complete problems and introduced a rapid randomised restarts strategy aimed at avoiding the long tails. More recently, Gomes et al. [8] studied the transition between heavy-tailed and non-heavy-tailed behaviour in runtime distributions for random problems and have characterised when the phenomenon occurs and when it does not.

The motivation behind the work we present in this paper comes from a number of interesting observations we made while reproducing the random problem experiments presented by Gomes et al., particularly while studying the heavy tail behaviour in the context of MAC [19]. We observed that once we enhanced MAC with any of the well-known standard variable ordering heuristics, such as min-domain [11], max-degree, min-dom/ddeg<sup>1</sup> [3], brelaz [5] and min-dom/wdeg [4, 17], heavy tails cannot be observed even for problems with 100 variables, for any density and tightness setting. Moreover, for the random problems used by Gomes et al. in their experiments with chronological backtrack search, we no longer observed heavy tails when we used min-dom/wdeg.

Some have conjectured that heavy-tailed behaviour is largely due to the characteristics of the algorithm used to solve the problems, in particular that the search algorithm makes a mistake that results in a significant amount of work being required to recover from it. In this paper we attempt to explore this hypothesis in detail. We show how variable and value ordering heuristics impact the heavy-tailed phenomenon one observes in the runtime distributions of backtrack search procedures. Our analysis focuses on the MAC algorithm while solving a large ensemble of satisfiable instances of the Quasi-groups with Holes (QWH) problem, encoded as a binary CSP. We combine different variable and value ordering heuristics with MAC to obtain a suite of algorithms that we can study. Our approach is based on analysing the refutations of insoluble (sub)trees encountered by MAC as it finds the first solution [13].

In this paper we present the following observations and results:

1. We observe that heavy-tailed behaviour associated with the runtime distribution of MAC on QWH of order 10 (QWH-10) can be eliminated by carefully selecting the search heuristics.
2. We perform a more detailed analysis of the effects that variable and value orderings have on heavy-tailedness. We show how combinations of variable and value ordering heuristics can result in a problem being *inherently heavy-tailed*. Specifically, we show that even if we were able use an Oracle to refute insoluble subtrees optimally, for some combinations of heuristics we would still observe heavy-tailed behaviour.
3. Finally, we study the distribution of refutations found using different combinations of heuristics and gain some further insights into what characteristics tend to give rise to heavy-tailed behaviour. Such a detailed analysis is the first of its kind to be reported in the literature.

---

<sup>1</sup> In this paper we abbreviate dynamic-degree as ‘ddeg’ and weighted-degree as ‘wdeg’.

## 2 Motivation and Summary

Two important questions one can attempt to address when studying heavy-tailed behaviour are *when* does the phenomenon occur, and *why*. Gomes et al. [8] have characterised *when* the phenomenon occurs, in this paper we focus on studying the reasons *why*.

Our work is motivated by the observation that even when using chronological backtracking, for certain classes of problems heavy tails can be eliminated by using carefully chosen search heuristics. Runtime distributions are sometimes characterised by long tails, or *heavy tails*, and are generally modelled using the expression  $1 - F(x) = P\{X > x\} \sim Cx^{-\alpha}, x > 0$ , where  $F(x)$  is the cumulative distribution function (CDF) of a probability distribution function  $f(x)$  (PDF), and  $C$  and  $\alpha$  are constants with  $\alpha \in (0, 2)$  and  $C > 0$ . A near-straight line in a log-log plot for  $1 - F(x)$ , with a slope equal to  $-\alpha$ , is a clear sign of heavy-tailed behaviour. For example, in Figure 1(a) we present results similar to those presented by Gomes et al. [8]. In this figure we use a chronological backtrack search procedure that uses both random variable and value orderings, solving problems with different levels of constrainedness (Model B; instances with 17 variables, 8 values, density 0.84, and tightness between 0.015 and 0.25, the point where the phase transition occurs). This figure shows that heavy-tailed behaviour can be observed in problems that are in the easy region, far from the phase transition, but are not necessarily trivial. However, as we approach the phase transition such behaviour disappears as instances become uniformly difficult.

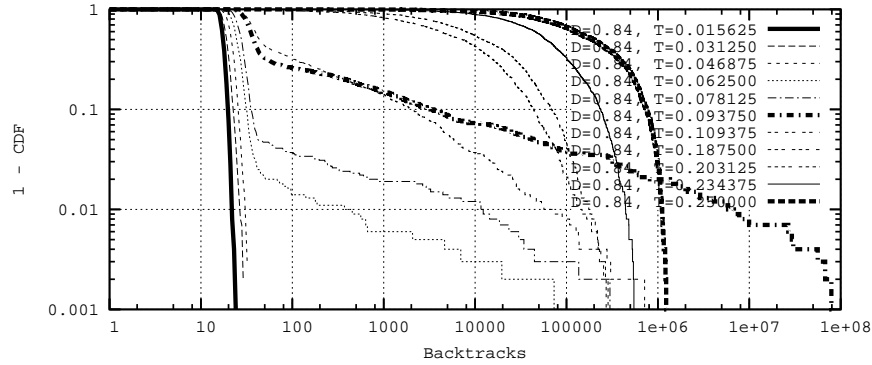
In Figure 1(b), we present results for the same problems, but with the variable ordering changed to min-dom/wdeg [4, 17]. Note that this change is sufficient to eliminate heavy tails from these problems since no straight line can be observed for any tightness. Clearly, the change in variable ordering had a dramatic impact on the runtime distribution.

A common intuitive understanding of the extreme variability of the runtime of backtracking is that the search procedure sometimes must refute a very large inconsistent subtree, causing considerable “thrashing”. In our efforts to gain an understanding of this intuition, we consider how search ordering affects the runtime distribution of MAC, rather than chronological backtracking, as it is one of the most commonly used algorithms in constraint satisfaction. We study its runtime distributions over many instances of QWH-10 for several configurations of the algorithm. By changing the variable and value ordering heuristics used, we vary MAC’s “quality”, essentially creating different algorithms that we can use for our investigation. As mentioned earlier, we used the following well-known *variable ordering heuristics*: min-domain [11], max-degree, min-dom/ddeg [3], brelaz [5] and min-dom/wdeg [4, 17]. As *value ordering heuristics*, we used max-conflicts, random and min-conflicts.

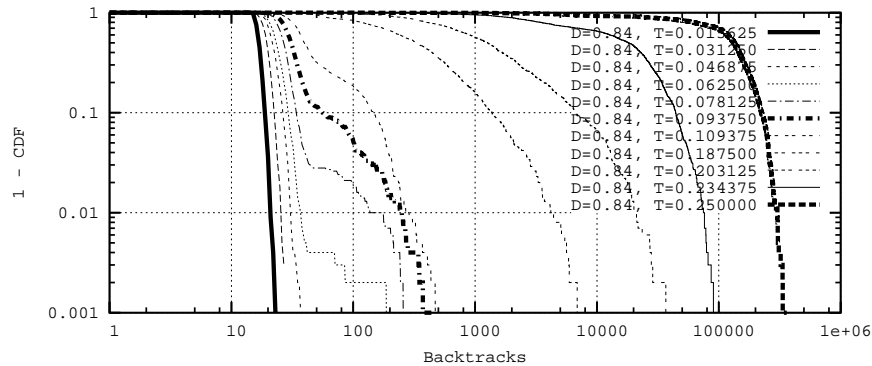
In Figure 2 we present MAC’s runtime distributions when solving instances of QWH-10 with different variable and value ordering heuristics<sup>2</sup>. When using a random value ordering heuristic, as presented in Figure 2(a), we observe heavy-tailed behaviour for

---

<sup>2</sup> In Figure 2, and in the remainder of the paper, we measure search effort in terms of “refutation sizes”, as this allows us to compare against search effort measured in terms of “optimal refutation sizes”. Refutations are introduced in Section 3.



(a) random variable and value orderings.

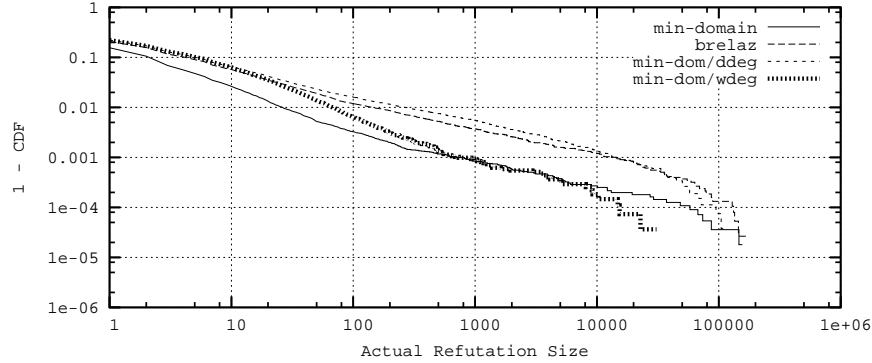


(b) min-dom/wdeg variable and random value orderings. Since chronological backtracking does no propagation, in this case min-dom/wdeg is effectively max-wdeg.

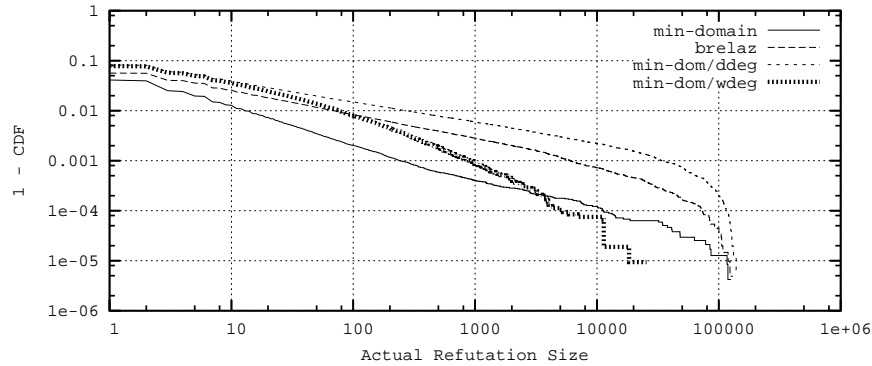
Fig. 1: The effect of the variable ordering on the cumulative distribution function of backtracks in random problems using chronological backtracking. Problems instances are from a Model B generator: 17 variables, 8 values, density 0.84, various tightness settings.

every variable ordering heuristic studied. However, when we replace the value ordering with min-conflicts, as presented in Figure 2(b), we no longer observe heavy tails when using the min-dom/wdeg variable ordering. Instead, for this configuration, we can see that the runtime distribution has become non-heavy-tailed, characterised by a curve in the log-log plot. Clearly, ordering heuristics also make a significant difference on this problem's runtime distribution.

As we will show later in this paper, there are a number of factors at play that explain the effects that ordering heuristics have on heavy-tailed behaviour.



(a) random value ordering heuristic and various variable orderings.



(b) min-conflicts value ordering heuristic and various variable orderings.

Fig. 2: Cumulative distribution function of the actual refutation size for MAC solving QWH-10 with different variable and value ordering heuristics. Note that when using a min-conflicts value ordering and a min-dom/wdeg variable ordering we no longer observe heavy tails (Figure 2(b)).

### 3 Definitions and Problems

**Definition 1 (Binary Constraint Satisfaction Problem).** We define a binary CSP as a 3-tuple  $P \hat{=} \langle V, D, C \rangle$  where  $V$  is a finite set of  $n$  variables  $V \hat{=} \{V_1, \dots, V_n\}$ ,  $D$  is a set of finite domains  $D \hat{=} \{D(V_1), \dots, D(V_n)\}$  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  $A_{ik} \hat{=} \langle V_i = v_k \rangle$  represents

a reduction of  $D(V_i)$  to  $\{v_k\} \subseteq D(V_i)$ . A solution to  $P$  is a set of distinct assignments  $\mathcal{S} \doteq \{A_{l_1 k_1}, \dots, A_{l_n k_n} \mid (v_{k_i}, v_{k_j}) \in C_{ij}\}$ .

**Definition 2 (Search Algorithm).** A search algorithm  $\Theta \doteq \langle \Lambda, \Delta, \prec_V, \prec_v \rangle$  is a combination of a branching method  $\Lambda$ , a consistency enforcement method  $\Delta$ , 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,  $\mathcal{N} \doteq \{A_{l_1 k_1}, \dots, A_{l_{p-1} k_{p-1}}, A_{l_p k_p}\}$ , totally ordered by a dynamic variable ordering heuristic  $\prec_V$ . The root of the search tree is a special node  $\mathcal{R} \doteq \emptyset$ . 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  $\Theta$ , a one-to-one mapping exists between the nodes in the search tree  $\mathcal{T}$  and the assignments made by  $\Theta$ .

### 3.1 Mistake Points and Refutations

To study the effects that variable and value ordering heuristics have on heavy-tailedness, we focus on the refutations of the mistakes made by each algorithm studied.

**Definition 4 (Mistake Point).** For a soluble problem  $P$ , a mistake point  $\mathcal{M}$  is a node identified by a set of assignments  $\mathcal{M} \doteq \{A_{l_1 k_1}, \dots, A_{l_{p-1} k_{p-1}}, A_{l_p k_p}\}$ , totally ordered by  $\prec_V$ , for which  $\mathcal{M} \setminus \{A_{l_p k_p}\}$  can be extended to a solution, but  $\mathcal{M}$  cannot. 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 upon the variable ordering used.

For a soluble problem  $P$ , let  $P_{\mathcal{M}} \doteq \{V_{\mathcal{M}}, D_{\mathcal{M}}, C_{\mathcal{M}}\}$  be the insoluble (sub)problem corresponding to  $\mathcal{M}$ , where  $V_{\mathcal{M}} \doteq V \setminus \{V_{l_1}, \dots, V_{l_p}\}$ ,  $C_{\mathcal{M}} \doteq \{C_{ij} \mid V_i, V_j \in V_{\mathcal{M}}, C_{ij} \in C\}$ , 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  $\mathcal{M} \doteq \emptyset$  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  $\Theta$ , 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 [18] for consistency enforcement and selects values randomly. Also, our version of MAC employs k-way branching [22], rather than binary branching, so that selecting a variable  $V_i$  creates  $|D(V_i)|$  branches in the search tree. 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  $\overline{\prec_V}$  s.t.  $|\mathcal{J}_{\mathcal{M}}|$  is minimised.

Sometimes, when the optimal refutation is hard to find, we compute the *quasi-optimal refutation*, defined as the smallest refutation whose height does not exceed that of the actual refutation. 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  $P_{\mathcal{M}}$ . Our experiments show that it is very rare that the quasi-optimal refutation is larger than the optimal (see Table 1). By accepting quasi-optimality, we can use the height of the actual refutation as an upper bound on the height of the optimal one, dramatically speeding-up the search for better refutations.

When searching for the quasi-optimal refutation it is oftentimes useful to compute a lower bound on its size. Firstly, this can improve the performance of the search since the search can be stopped when (and if) the smallest refutation found so far reaches that lower bound. Secondly, the lower bound can be plotted alongside with the quasi-optimal refutation size to visually reduce the factor of uncertainty introduced by quasi-optimality. Optimal refutations may, in theory, be smaller than quasi-optimal refutations, but they cannot be smaller than the lower bound.

**Definition 6 (Refutation Size Lower Bound).** *An  $n$ -level lower bound for a refutation corresponding to a mistake point is the minimum number of nodes that an  $n$ -level look-ahead determines any refutation for that particular (sub)tree must have.*

A lower bound can be computed using the look-ahead method described in [13]. Such a lower bound is quite conservative, as there is absolutely no guarantee that an optimal refutation of that size exists, although that is often the case with QWH-10. We briefly review the look-ahead methodology here.

Whenever a variable  $V_i$  is selected at a certain level 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.

Clearly, the further we look ahead, the less conservative the lower bound. However, look-ahead levels greater than 2 tend to be very time consuming and, therefore, the amount of look-ahead to use must be experimentally determined for each problem class being studied. We have concluded that for the class of quasigroup problems used in this paper, a look-ahead level of 3 was the most appropriate when computing the lower bound, and a look-ahead level of 1 was the most appropriate for the upper bound.

### 3.2 Problem Domain

As mentioned before, our experiments were focused around quasigroup completion problems. We briefly introduce them here.

**Definition 7 (Quasigroup Completion Problems).** *A quasigroup is a set  $Q$  with a binary operation  $\star : Q \times Q \rightarrow Q$ , such that for all  $a$  and  $b$  in  $Q$  there exist unique elements in  $Q$  such that  $a \star x = b$  and  $y \star a = b$ . The cardinality of the set,  $n = |Q|$ , is called the order of the quasigroup.*

A quasigroup can be viewed as an  $n \times n$  multiplication table defining a Latin square, which must be filled with unique integers on each row and column. The Quasigroup Completion Problem (QCP) is the problem of completing a partially filled Latin square. Quasigroup with Holes (QWH) are satisfiable instances obtained by starting with a complete Latin square and unassigning a number of cells according to the Markov chain Monte Carlo approach proposed by Jacobson and Matthews in [14]. QWH problem instances are considerably harder when the distribution of the holes is balanced, i.e, when the number of unassigned cells is approximately the same across the different rows and columns [1, 14, 15].

Originally just mathematical curiosities, Latin squares have found practical applications in many scientific and engineering fields such as statistics, scheduling, drug tests design, and cryptography. Quasigroup problems have a small-world topology [23] and are known to be NP-complete [6]. QWH problems pre-assign a percentage of the cells in the table, introducing perturbations into the structure of the constraint network and bringing problems closer to real-world instances.

## 4 Finding Optimal Refutations

We introduced in [13] an algorithm for obtaining optimal refutations for binary CSPs. In this paper we significantly improved that algorithm's efficiency so that it can tackle larger problems. All the optimisations described here are general in nature, i.e. they can be applied to any class of problems, but proved particularly useful in dealing with QWH-10 problems due to their relatively shallow refutations. We observed that for the

problems under consideration, while most actual refutations have heights up to 30, early experiments showed that most optimal refutations have heights below 5. The nature of the search for optimal refutations is such that the branching factor is significantly larger than that of a normal search tree, and we estimate that searching for an optimal refutation of height 6 for an instance of a QWH-10 problem could take several months to complete on a Pentium M CPU. Consequently, limiting the height of the refutations considered can dramatically improve efficiency.

An important design decision in the optimal refutation search algorithm was to use an iterative-deepening strategy [16]. The algorithm starts off by searching for refutations of height 1, then for refutations of height 2, and so on, until it reaches a height equal to either the number of variables in the (sub)problem, or the size of the smallest refutation found up until that point. The motivation behind using an iterative-deepening strategy is based on the expectation that small refutations are likely to have smaller heights than larger refutations, an expectation that has been validated by our experiments. The successive expansions of the search horizon can increase the likelihood of finding earlier refutations that are better than the actual, thus lowering the current refutation size upper bound and significantly speeding up the search in the rest of the tree.

Our improved algorithm uses the refutation size lower bound to limit the height in the iterative-deepening loop. Consider an example where we use  $m$  levels of look-ahead and obtained a lower bound of  $l$  nodes. The number of nodes in excess of  $m$  that would be part of any refutation of height  $m$  is  $e = l - m$ . If the best refutation found up until this point is of size  $r$ , then we can immediately conclude that since *any* refutation of height  $d \geq m$  would contain at least  $e$  additional nodes, searching for refutations of heights greater than  $r - e$  cannot produce a smaller refutation. Moreover, updating the lower bound as we search deeper into the tree allows us to stop the search as soon as it discovers a refutation whose size equals the lower bound (this seems to occur quite frequently). Finally, once the algorithm completes the search for refutations of size  $r - e$ , we know that the best refutation found is *optimal*.

The number of uninstantiated variables involved in each refutation is indirectly a factor affecting the height limit in the iterative-deepening loop. We have modified MAC so that, before selecting a new variable, it automatically marks as *instantiated* every variable whose domain has been reduced to a single value as a result of restoring arc-consistency. This reduces the height of the search tree, avoids unnecessary backtracking over singletons, and makes sure that such variables do not take part in any refutation. Mistake trees thus involve a smaller number of variables and are easier to deal with.

Despite of all these optimisations, our data-set contains a small number of instances (23 out of over 1,000,000) for which the search for the optimal refutation timed out after 2 hours for at least one mistake point, and in some of these cases no *improved* refutation was found. For some of these instances we succeeded in finding improved refutations by using a rapid random restarts strategy [9]. For others, we employed a certain level of *optimism*, i.e. we tried at each level a limited number of variables, in effect searching only what appeared to be the most promising area of the search space. The shorter-than-actual refutations that we found using these two methods are not known to be optimal or quasi-optimal, yet they are significantly smaller than their corresponding

actual refutations, and by finding them we avoided incorrectly elongating the tails of the plots.

## 5 Experiments

Our experiments<sup>3</sup> were performed on satisfiable QWH-10 problem instances (100 variables) with 90% random balanced holes<sup>4</sup>, encoded as binary CSPs. We aimed to study the relationship between actual and optimal refutations, as well as the way they evolve as better and better search algorithms are used to solve a large set of instances.

Our empirical study included 4 variable ordering heuristics: brelaz, min-domain, min-dom/ddeg and min-domain/wdeg, and 3 value ordering heuristics: random, min-conflicts<sup>5</sup> and its anti-heuristic, max-conflicts. We always broke ties randomly. QWH-10 instances are too difficult to solve using random variable orderings or variable ordering anti-heuristics, which is why these heuristics could not be included.

Using a Beowulf cluster of 32 CPUs over a period of 6 weeks we accumulated experimental data on all the 12 variations of MAC, totaling over 1,000,000 instances. Our intention was to avoid running an artificially randomised algorithm multiple times on the same instance. We computed actual refutations, lower bounds, and attempted to compute optimal refutations, reporting the cumulative size of each for every instance. While in the vast majority of cases we did find the optimal refutations, we encountered some instances for which we could only find improved refutations (i.e. refutations for which we were not able to guarantee quasi-optimality), and some for which the search timed out without finding any improved refutation. Table 1 gives the actual percentages.

Table 1: Optimality (%optimal / %quasi-optimal / %improved / %timed out).

	max-conflicts	random	min-conflicts
<b>min-domain</b>	99.37 / 0.19 / 0.40 / 0.03	99.44 / 0.25 / 0.31 / 0.00	97.96 / 0.74 / 1.28 / 0.02
<b>min-dom/ddeg</b>	95.41 / 4.01 / 0.58 / 0.01	98.27 / 0.42 / 1.30 / 0.01	86.67 / 3.37 / 9.84 / 0.11
<b>brelez</b>	99.24 / 0.19 / 0.53 / 0.03	98.69 / 0.33 / 0.97 / 0.01	87.22 / 3.10 / 9.47 / 0.22
<b>min-dom/wdeg</b>	99.26 / 0.38 / 0.36 / 0.00	97.97 / 0.81 / 1.22 / 0.01	86.64 / 3.70 / 9.33 / 0.33

Figure 3 includes results for each of our 12 experiments and shows the effects of the various heuristics on the shape of the refutation size cumulative distribution function. The plots are organised roughly in increasing order of efficiency from left to right and from top to bottom. It is important to point out that the shorter refutation and lower bound plots are specific to each search algorithm, simply because different algorithms make mistakes in different places. The lower bound is plotted to give an absolute mini-

<sup>3</sup> Code freely available with source at <http://hulubei.net/tudor/csp>.

<sup>4</sup> Generated using code based on Carla Gomes' *Isencode* quasigroup generator.

<sup>5</sup> This heuristic selects the value that is inconsistent with the smallest number of other values in the domains of neighbouring variables.

mum on the size of the refutations even for those instances where we could not find the optimal or quasi-optimal refutations (see Table 1).

We notice from the first column of Figure 3 that a search algorithm employing a poor value ordering heuristic (max-conflicts) always exhibits heavy tails, irrespective of which one of the 4 variable ordering heuristics we use. Moreover, heavy tails still exist, albeit with a different slope, even if, once a mistake has been made, the search algorithms were to use an Oracle that could provide the shortest refutation for that mistake. In other words, in such cases the runtime distribution of an algorithm would be *guaranteed to exhibit heavy-tailed behaviour*.

We increase the quality of the value ordering heuristic by switching from max-conflicts to a random ordering and observe that while the actual refutations remain heavy-tailed, as we improve the quality of the variable ordering, the *shorter* refutations and the lower bounds start becoming less and less heavy-tailed. For the lack of a better term, we call such plots *borderline heavy-tailed*. This phenomenon becomes even more pronounced as we move over to min-conflicts. In that case, all lower bounds become non-heavy-tailed, and with the exception of min-domain, shorter refutations become borderline heavy-tailed. Finally, the best combination of heuristics, min-conflicts + min-dom/wdeg, succeeds at eliminating heavy tails even from the actual refutations, while at the same time keeping the actual refutations much closer to their corresponding optimal than any of the other 11 variations of MAC<sup>6</sup>.

We observe in Figure 4 several factors contributing to the behaviour of our best performing algorithm. Firstly, for QWH-10, algorithms using the min-domain, min-dom/ddeg, and brelaz variable orderings encounter similarly large maximum refutations *regardless* of the value ordering used. Secondly, any algorithm using min-dom/wdeg encounters a maximum refutation size that is a factor of 5 smaller than the maximum encountered by algorithms using the other variable ordering heuristics. Thirdly, improving the value ordering heuristic from max-conflicts or random to min-conflicts results in a decrease in the probability of each non-trivial<sup>7</sup> refutation size occurring.

However, it is interesting to see in Figure 5 that any algorithm using min-dom/wdeg, while not encountering extremely large refutations, tends to have a higher probability of encountering all other sized non-trivial refutations, i.e. over the range of refutation sizes it encounters, it does worse than any other variable ordering for all value orderings.

Therefore, to summarise, the combination of the factors outlined above seems to eliminate heavy tails from the QWH-10 problems we have studied, despite the fact that neither min-conflicts nor min-dom/wdeg alone seems to be capable of doing that. This is a somewhat more complex scenario that one might have initially envisaged.

Finally, the inherent heavy-tailedness that we observe suggests that while recovering from failure is important (which is typically the focus of research on variable orderings), we should also focus considerable attention on the interrelationship between variable selection and value selection in order to mitigate heavy-tailed runtime distributions by failing less.

---

<sup>6</sup> Table 1 shows that for this combination of heuristics we found the lowest percentage of optimal refutations, so the true runtime distribution can only be even more obviously non-heavy-tailed.

<sup>7</sup> Obviously, this implies an increase in the probability of the occurrence of trivial mistakes, i.e. those that can be refuted by propagation alone.

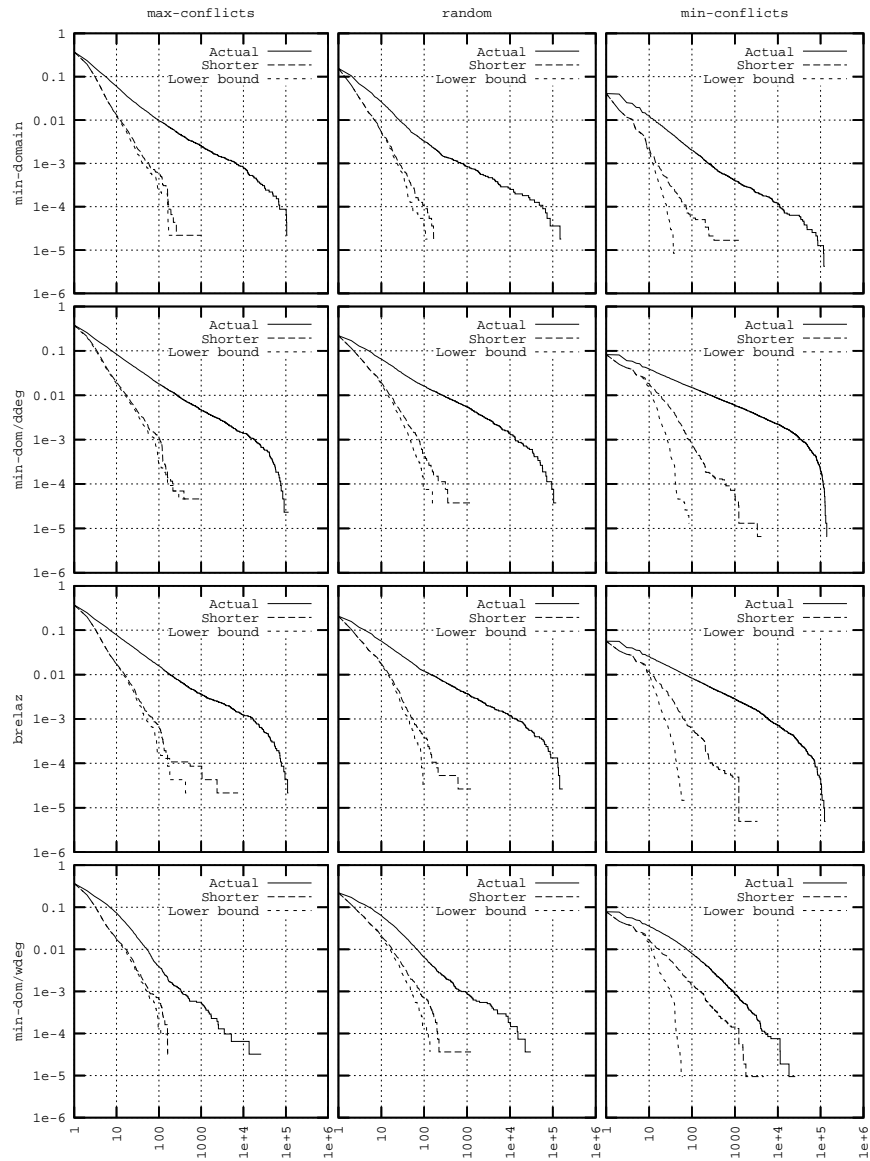
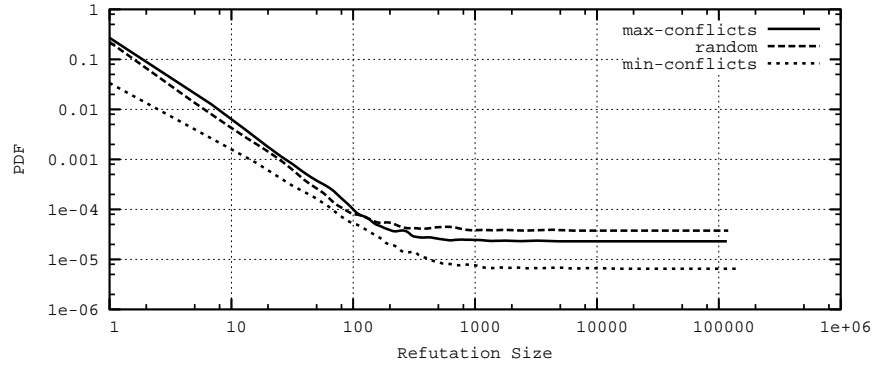
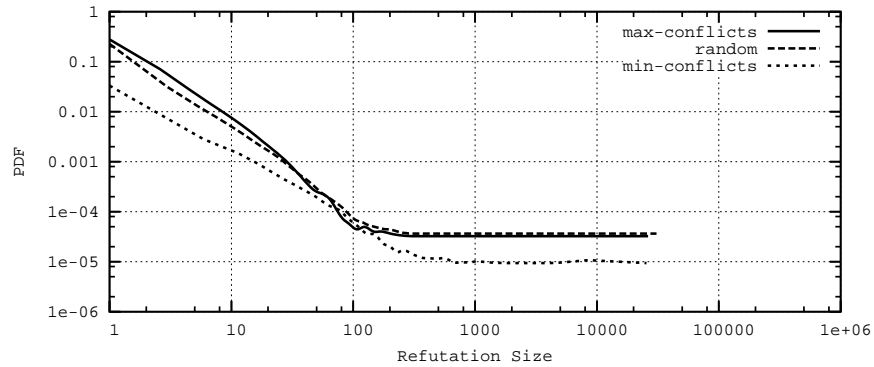


Fig. 3: Cumulative distribution function of the refutation size for QWH-10 problems with 90% holes. We vary the value ordering across columns and variable ordering across rows. Shorter refutations are either optimal, quasi-optimal, or simply the shortest improved refutations we could find that were smaller than the corresponding actual refutations.



(a) Variable ordering: min-dom/ddeg (very similar results for min-domain and brelaz).

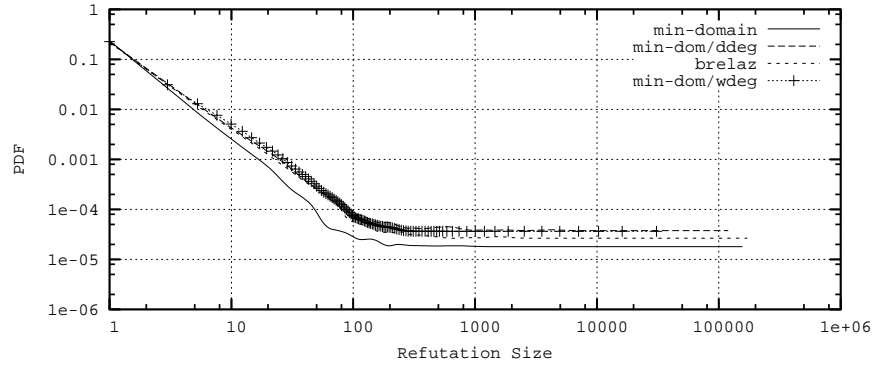


(b) Variable ordering: min-dom/wdeg.

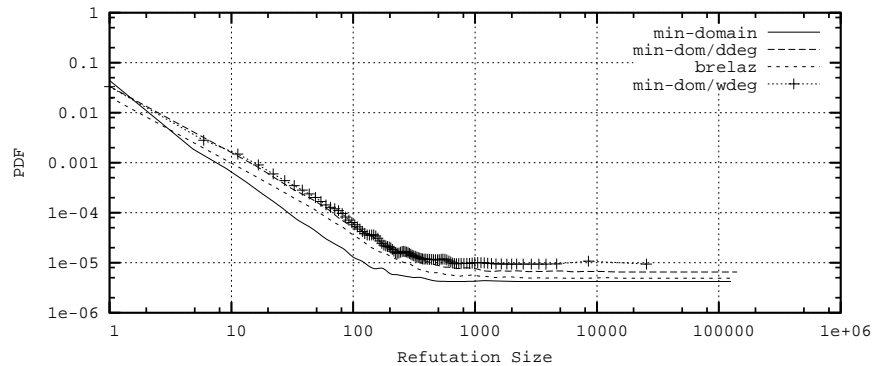
Fig. 4: Probability distribution function for the actual refutation sizes obtained with various search algorithms, grouped by variable ordering.

## 6 Conclusions

Much progress has been made on understanding problem hardness, typical-case complexity and the behaviour of backtrack search. The heavy-tailed phenomenon that characterises the runtime distributions of backtrack search procedures has received considerable attention. We have shown that a good choice of variable and value orderings can have a dramatic impact on the runtime distribution. A good combination can eliminate heavy-tailed behaviour from certain classes of problems, while a poor choice not only ensures that such behaviour is observed, but also that the nature of the insoluble subtrees encountered guarantees that this is the case.



(a) Value ordering: random (very similar results for max-conflicts).



(b) Value ordering: min-conflicts.

Fig. 5: Probability distribution function for the actual refutation sizes obtained with various search algorithms, grouped by value ordering.

We believe our work provides motivation for more focused research on the interplay between variable and value selection during search. We also believe that there are many directions that one can follow with respect to the utility of empirically studying optimal refutations of insoluble subtrees in relation to runtime distributions. We intend to explore these opportunities in more detail as part of our future work.

## Acknowledgments

This material is based on work supported by Science Foundation Ireland under Grant 00/PI.1/C075. We would like to thank Mark Hennessy, Barbara Smith and Tom Carchrae

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

- [1] D. Achlioptas, C.P. Gomes, H.A. Kautz, and B. Selman. Generating satisfiable problem instances. In *Proceedings of AAAI-2000*, pages 256–261, 2000.
- [2] C. Bessière, C. Fernández, C.P. Gomes, and M. Valls. Pareto-like distributions in random binary csp. In *Proceedings of ACIA-2003*, 2004.
- [3] 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.
- [4] F. Boussemart, F. Hemery, C. Lecoutre, and L. Sais. Boosting systematic search by weighting constraints. In *Proceedings of ECAI-2004*, pages 146–150, 2004.
- [5] D. Brélez. New methods to color the vertices of a graph. *Communications of the ACM*, 22(4):251–256, 1979.
- [6] C. Colbourn. Embedding partial steiner triple systems is NP-complete. *Combinatorial Theory*, A(35):100–105, 1983.
- [7] I.P. Gent and T. Walsh. Easy problems are sometimes hard. *Artificial Intelligence*, 70:335–345, 1994.
- [8] 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.
- [9] C.P. Gomes, B. Selman, and N. Crato. Heavy-tailed distributions in combinatorial search. In *Proceedings of CP-1997*, pages 121–135, 1997.
- [10] 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.
- [11] R.M. Haralick and G.L. Elliott. Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence*, 14(3):263–313, 1980.
- [12] T. Hogg and C.P. Williams. The hardest constraint problems: A double phase transition. *Artificial Intelligence*, 69:359–377, 1994.
- [13] T. Hulubei and B. O’Sullivan. Optimal refutations for constraint satisfaction problems. In *Proceedings of IJCAI-2005*, 2005.
- [14] M.T. Jacobson and P. Matthews. Generating uniformly distributed random latin squares. *Combinatorial Design*, 4:405–437, 1996.
- [15] H.A. Kautz, Y. Ruan, D. Achlioptas, C.P. Gomes, B. Selman, and M.E. Stickel. Balance and filtering in structured satisfiable problems. In *Proceedings of IJCAI-2001*, pages 351–358, 2001.
- [16] R.E. Korf. Depth-first iterative-deepening: an optimal admissible tree search. *Artificial Intelligence*, 27(1):97–109, 1985.
- [17] C. Lecoutre, F. Boussemart, and F. Hemery. Backjump-based techniques versus conflict-directed heuristics. In *Proceedings of ICTAI-2004*, pages 549–557, 2004.
- [18] A.K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–118, 1977.
- [19] D. Sabin and E.C. Freuder. Contradicting conventional wisdom in constraint satisfaction. In *Proceedings of ECAI-1994*, pages 125–129, 1994.
- [20] B.M. Smith. In search of exceptionally difficult constraint satisfaction problems. In *Constraint Processing, Selected Papers*, LNCS 923, pages 139–156, 1995.
- [21] B.M. Smith and S. Grant. Sparse constraint graphs and exceptionally hard problems. In *Proceedings of IJCAI-1995*, pages 646–651, 1995.
- [22] 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.
- [23] T. Walsh. Search in a small world. In *Proceedings of IJCAI-1999*, pages 1172–1177, 1999.