Fail First Heuristics: An Update

J. Christopher Beck[1] and **Patrick Prosser**[2] and **Richard J. Wallace**[1]

1 Introduction

In ECAI 1998 Smith & Grant performed a study [4] of the fail-first principle of Haralick & Elliott [2]. The fail-first principle states that "To succeed, try first where you are most likely to fail." The basic hypothesis of Smith & Grant was that if failing first is such a good thing then more of it must be better. Therefore, creating heuristics with a stronger ability to fail early should increase search efficiency. Efficiency was defined as the number of constraint checks required to find a solution to a problem or to prove that no solution exists. Because the focus of the study was on the relation between fail-firstness and search efficiency, the computational effort to make those heuristic decisions was (correctly) factored out of the experiments.

Experiments were performed over randomly generated binary constraint satisfaction problems. Each set of problems was defined by a 4-tuple $\langle n, m, p_1, p_2 \rangle$, where n is the number of variables, m is the uniform domain size, p_1 is the proportion of edges in the constraint graph, and p_2 is the uniform constraint tightness. p_2 can be considered as the probability that when a pair of constrained variables are instantiated they will be in conflict. All their experiments were over problems with $n = 20$ and $m = 10$.

Using the forward checking algorithm [2] and standard chronological backtracking Smith & Grant tested four heuristics engineered for increasing levels of fail-firstness.

- **FF**: Choose the variable with the smallest remaining domain.
- **FF2**: The variable, v_i, chosen is the one that maximizes $(1 - (1 - p_2^m)^{d_i})^{m_i}$, where m_i is the current domain size of v_i, and d_i is the future degree of v_i. The FF2 heuristic takes into account an estimate (based on the initial parameters of problem generation) of the extent to which each value of v_i is likely to be consistent with the future variables of v_i.
- **FF3**: FF3 builds on FF2 by using the current domain size of future variables rather than m. The variable, v_i, chosen is the one that maximizes the expression (1) below, where C is the set of all constraints in the problem, F is the set of unassigned variables, and $P = p_2$.
- **FF4**: FF4 modifies FF3 by substituting the current tightness, $P = p_{ij}$, of the future constraints (the fraction of tuples from the cross-product of the current domains that fail to satisfy the constraint) instead of p_2.

$$(1 - \prod_{(v_i,v_j) \in C, v_j \in F} (1 - P^{m_j}))^{m_i} \qquad (1)$$

[1] Cork Constraint Computation Center Computing Science Department, University College Cork, Ireland, {c.beck,r.wallace}@4c.ucc.ie

[2] Department of Computing Science, University of Glasgow, Scotland, pat@dcs.gla.ac.uk

The progression from FF through to FF4 uses more and more information in determining which variable is most likely to fail at any given stage of search. With respect to total search effort, Smith & Grant expected that the heuristics would be ranked as follows: FF $>$ FF2 $>$ FF3 $>$ FF4, where $>$ means "results in greater search effort than". Their results were not as expected. Through a number of experiments Smith & Grant showed that except on easy problems, FF2 incurred the least search effort, followed by FF3, FF and finally FF4. That is, they observed the following order: FF4 $>$ FF $>$ FF3 $>$ FF2. This ordering is clearly at odds with the hypothesis of a simple mapping between the ability to fail-first and search effort. Smith & Grant concluded that there must be some other factor at work in determining the search efficiency for variable ordering heuristics.

2 A Bug in FF4

In order to continue on from Smith & Grant's work and investigate additional factors that may impact the search efficiency of variable ordering heuristics, our first step was to reproduce their experiments. We were able to reproduce the results for FF, FF2, and FF3, but we were unable to reproduce the behavior of FF4. Our version of FF4 was orders of magnitude *better* than the FF4 reported by Smith & Grant. Our results were tested on a number of independent implementations of the algorithms and problem generators: a solver written in C, one in Java, one based on ILOG Solver, and one written in LISP. All of our attempts failed to reproduce the reported results.

Barbara Smith and Stuart Grant kindly provided us with the source code of their solver and we found a programming error in their code. When computing the value of p_{ij} for the FF4 heuristic they used integer division instead of floating point division. This resulted in p_{ij} being assigned a value of 1 for a variable that is inconsistent with an adjacent variable and 0 otherwise. Therefore the search algorithm will either select a future variable that is arc-inconsistent with some other future variable, generating a dead-end, or select any other variable with equal likelihood.

When this error was corrected in their solver we were able to reproduce *our* results using *their* solver. Figure 1 presents the median number of constraint checks to find a solution or to prove that no solution exists for problems from the set $\langle 20, 10, 0.2 \rangle$. The constraint tightness was varied from 0.01 to 0.99 in steps of 0.01. For each combination of parameters 1000 problem instances are generated. Median checks are plotted against κ (kappa), the measure of constrainedness proposed by [1]. In the new results we typically see the following ordering of the heuristics: FF $>$ FF3 $>$ FF4 $\geq$ FF2. This ordering was reproduced over all values of p_1 in the range 0.1 to 1.0. Although we have fixed Smith and Grant's bug, resulting in a new ordering of the heuristics, we are still lead to the same conclusion: *trying harder to fail-first does not guarantee a reduction in search effort.*

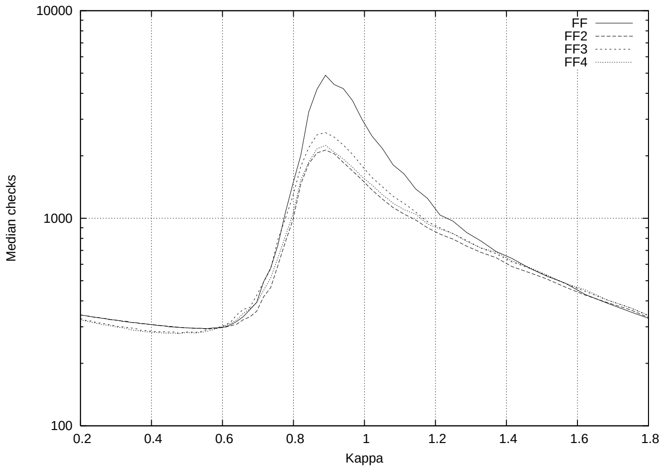

Figure 1. The median number of checks for the $\langle 20, 10, 0.2\rangle$ problem set. The heuristics are ranked as follows : FF $>$ FF3 $>$ FF4 $\geq$ FF2. That is, FF2 and FF4 compete as the best heuristics, and FF3 is better than FF.

3 Varying Problem Size

Do we get the same ranking of the heuristics FF $>$ FF3 $>$ FF4 $\geq$ FF2 when we look at larger problems? We generated random problems of varying size, from $n = 20$ up to $n = 70$. These problems were generated such that in each set of problems variables had the same average degree of 10, i.e. each variable was on average constrained by 10 other variables. Variables had a uniform domain size of 10. Each set of problems was generated such that constrainedness $\kappa \approx 0.9$. Figure 2 gives contours for the four heuristics, with median consistency checks plotted against problem size. We see that the ranking of the heuristics is not influenced by problem size. Problems with more than 60 variables were too hard for the FF heuristic. Figure 2 shows contours for the soluble problems only. Similar experiments were performed with $\kappa > 1.0$ such that all problems were insoluble. Again, we observed the same ranking of the heuristics. Therefore, as far as the relative ranking of our heuristics is concerned, size does not matter.

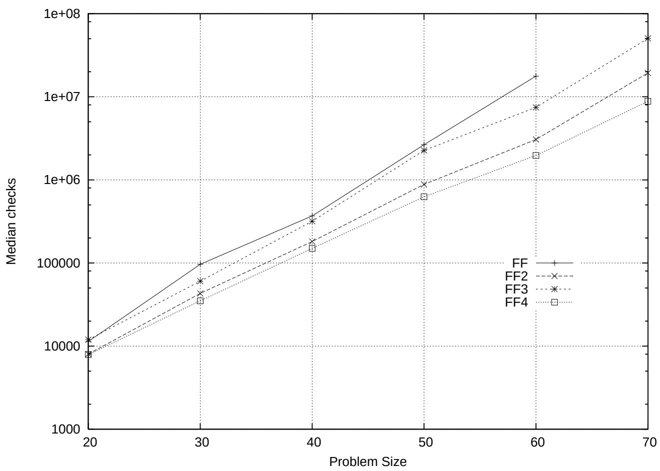

Figure 2. The median consistency checks for problems with 20 to 70 variables. All problems have 10 values per variable, a density that results in 10 constraints on each variable, and a tightness set so as to give a $\kappa \approx 0.9$. All problems are soluble and at each problem size we have 50 instances.

4 Moving to MAC

Might the consistency enforcement algorithm have an effect on the heuristics? To test this out we repeated the above experiments, but this time using the maintaining-arc consistency algorithm (MAC) [3]. As the name implies, whenever a variable is instantiated the future sub-problem is made arc-consistent. If this results in a domain wipe-out, a new value is tried, and failing that, backtracking takes place. Our results are presented in Figure 3. The ranking of the heuristics has changed! We now have the order FF $>$ FF2 $>$ FF3 $>$ FF4, suggesting that as we try harder to fail-first we do indeed see a reduction in search effort, but this is algorithm dependent. The results in Figure 3 are for soluble problems, and though not shown, we got the same ranking over insoluble problems. Again, these results were replicated across our solvers and across sites.

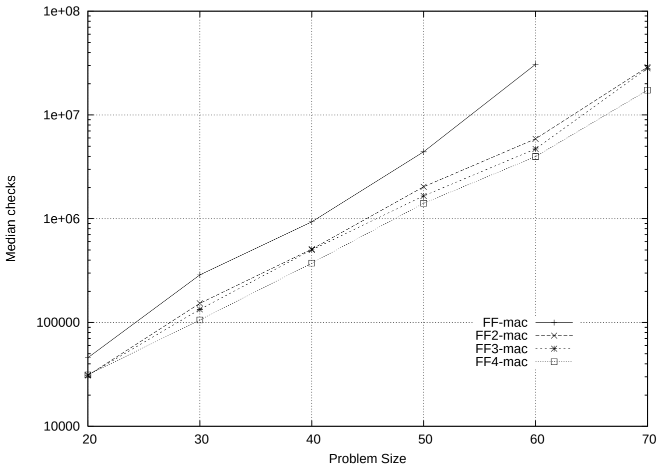

Figure 3. The median consistency checks for problems with 20 to 70 variables using MAC. All problems have 10 values per variable, a density that results in 10 constraints on each variable, and a tightness set so as to give a $\kappa \approx 0.9$. All problems are soluble and each problem size has 50 instances.

5 Conclusion

Smith & Grant claimed that trying harder to fail first does not result in reduced search effort and that there must be some other factor at work in determining the search efficiency for variable ordering heuristics. We have discovered a bug in their experiments, and have shown that their most aggressive failing-first heuristic FF4 ties with their best heuristic FF2. We have also shown that when we change algoritim, moving to MAC, trying harder to fail-first does indeed reduce search effort, i.e. this phenomenon is algorithm dependent. Work still needs to be done to explain why, when using forward checking, FF2 is as good as FF4, and more generally *what other factors than failing-first influence heuristic performance?*

REFERENCES

[1] Ian P. Gent, Ewan MacIntyre, Patrick Prosser, and Toby Walsh, 'The constrainedness of search', in *AAAI/IAAI, Vol. 1*, pp. 246–252, (1996).

[2] R. M. Haralick and G. L. Elliott, 'Increasing tree search efficiency for constraint satisfaction problems', *Artificial Intelligence*, **14**, 263–314, (1980).

[3] Daniel Sabin and Eugene Freuder, 'Contradicting Conventional Wisdom in Constraint Satisfaction', in *Eleventh European Conference on Artificial Intelligence (ECAI 94)*, pp. 125–129. John Wiley & Sons, Ltd., (1994).

[4] Barbara. M. Smith and Stuart. A. Grant, 'Trying harder to fail first', in *Thirteenth European Conference on Artificial Intelligence (ECAI 98)*, pp. 249–253. John Wiley & Sons, Ltd., (1998).