

Carla Gomes and Toby Walsh

Randomness and Structure

To appear in *Handbook of Constraint Programming*, Elsevier, 2006

Abstract:

This chapter covers research in constraint programming (CP) and related areas involving random problems. Such research has played a significant role in the development of more efficient and effective algorithms, as well as in understanding the source of hardness in solving combinatorially challenging problems.

Random problems have proved useful in a number of different ways. Firstly, they provide a relatively “unbiased” sample for benchmarking algorithms. In the early days of CP, many algorithms were compared using only a limited sample of problem instances. In some cases, this may have led to premature conclusions. Random problems, by comparison, permit algorithms to be tested on statistically significant samples of hard problems. However, as we outline in the rest of this chapter, there remain pitfalls waiting the unwary in their use. For example, random problems may not contain structures found in many real world problems, and these structures can make problems much easier or much harder to solve. As a second example, the process of generating random problems may itself be “flawed”, giving problem instances which are not, at least asymptotically, combinatorially hard.

Random problems have also provided insight into problem hardness. For example, the influential paper by Cheeseman, Kanefsky and Taylor highlighted the computational difficulty of problems which are on the “knife-edge” between satisfiability and unsatisfiability. There is even hope within certain quarters that random problems may be one of the links in resolving the $P=NP$ question.

Finally, insight into problem hardness provided by random problems has helped inform the design of better algorithms and heuristics. For example, the design of a number of branching heuristics for the Davis Logemann Loveland satisfiability (DPLL) procedure has been heavily influenced by the hardness of random problems. As a second example, the rapid randomization and restart (RRR) strategy was motivated by the heavy-tailed runtime distributions for backtracking style search procedures on random quasigroup completion problems.