Handbook of Constraint Programming
Edited by F. Rossi, P. van Beek and T. Walsh

Chapter 18

Randomness and Structure

Carla Gomes and Toby Walsh

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 lead 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 [12] highlighted the computational difficulty of problems which are on the "knife-edge" between satisfiability and unsatisfiability [84]. 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 [45, 44] was motivated by the discovery of heavy-tailed runtime distributions in backtracking style search procedures on random quasigroup completion problems.

18.1 Random Constraint Satisfaction

We begin by introducing the random problems classes studied in constraint satisfaction, and discussing various empirical and theoretical results surrounding them.

18.1.1 Models A to D

Most experimental and theoretical studies use one of four simple models of random constraint satisfaction problems. In each, we generate a constraint graph G, and then for each edge in this graph, we choose pairs of incompatible values for the associated conflict matrix. The models differ in how we generate the constraint graph and how we choose incompatible values. In each case, we can describe problems by the 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 a measure of the density of the constraint graph, and p_2 is a measure of the tightness of the constraints.

model A: we independently select each one of the $n(n-1)/2$ possible edges in G with probability p_1, and for each selected edge we pick each one of the m^2 possible pairs of values, independently with probability p_2, as being incompatible;

model B: we randomly select exactly $p_1 n(n-1)/2$ edges for G, and for each selected edge we randomly pick exactly $p_2 m^2$ pairs of values as incompatible;

model C: we select each one of the $n(n-1)/2$ possible edges in G independently with probability p_1, and for each selected edge we randomly pick exactly $p_2 m^2$ pairs of values as incompatible;

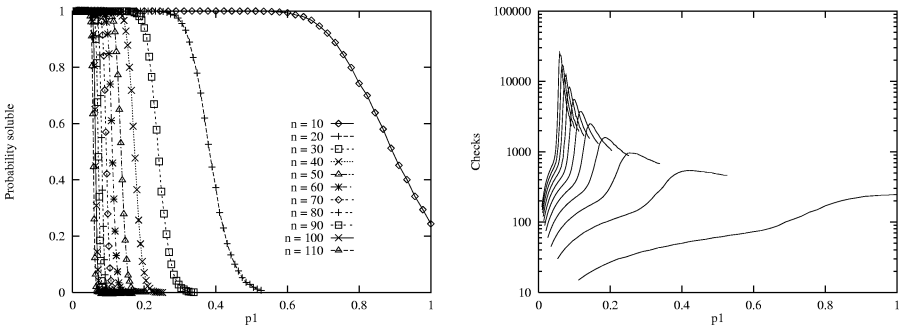
model D: we randomly select exactly $p_1 n(n-1)/2$ edges for G, and for each selected edge we pick each one of the m^2 possible pairs of values, independently with probability p_2, as being incompatible;

Whilst p_1 and p_2 can be either a probability or a fraction, similar results are observed with the four different models. Most experimental studies typically fix n and m, and vary p_1 and/or p_2. Typical parameter ranges include $\langle 10, 10, p_1, p_2 \rangle$, $\langle 20, 10, p_1, p_2 \rangle$, $\langle 10 - 200, 3, p_1, 1/9 \rangle$, and $\langle 10 - 200, 3, p_1, 2/9 \rangle$. The penultimate of these parameter ranges resembles graph 3-colouring. See Table 1 in [27] for a more extensive survey.

18.1.2 Phase Transition

Random problems generated in this way exhibit phase transition behaviour similar to that seen in statistical mechanics [12]. Loosely constrained problems are almost surely satisfiable. As we increase the parameters and constrain the problems more, problems become almost surely unsatisfiable. As n increase, the transition between satisfiable and unsatisfiable problems becomes sharper and sharper. In the limit, it is a step function [22]. Using a Markov first moment method, the location of this phase transition can be predicted to occur where the expected number of solutions is approximately 1 [73, 80]. Associated with this rapid transition in satisfiability of problems, is a peak in problem hardness for a wide range both of systematic and local search methods [12, 67, 73, 80]. Such problems are on the "knife-edge" between satisfiability and unsatisfiability [84]. It is very hard to tell if they are satisfiable or unsatisfiable. If we branch on a variable, the resulting subproblem is

smaller but otherwise tends to look similar. We can only determine if the current subproblem is satisfiable deep in the search tree. See Figure 18.1 for some graphs displaying the "easy-hard-easy" pattern associated with phase transitions.



(a) Probability (b) Median Checks

Figure 18.1: Phase transition for Model B problems with $\langle n, 3, p_1, 2/9\rangle$ (a) percentage satisfiability and (b) median search effort for FC-CBJ with the fail-first heuristic against p_1, and n from 10 to 110. Graphs taken from [27].

Whilst the hardest problems typically occur close to this rapid transition in satisfiability, hard problems can occur elsewhere. In particular, in the easy and satisfiable region, problems can occasionally be very hard to solve, especially for systematic search procedures like forward checking [32, 47, 46]. Such exceptionally hard problems (EHPs) appear to be a consequence of early branching mistakes. Better branching heuristics, more informed backtracking mechanisms, greater constraint propagation and restart strategies can all reduce the impact of EHPs greatly. Since curves of median search effort may disguise the appearance of EHPs, experimentalists are encouraged to look for outliers.

18.1.3 Constrainedness

Williams and Hogg introduced the first comprehensive theoretical model of such phase transition behaviour for constraint satisfaction problems [86]. More recently, Gent et al. presented a theory that works across a wide range of problems and complexity classes including constraint satisfaction and satisfiability problems [30]. This theory is based around the definition of the "constrainedness" of a problem using the parameter κ. For an ensemble of problems:

$$\kappa \quad = \quad 1 - \frac{\log_2(\langle Sol\rangle)}{N}$$

Where $\langle Sol\rangle$ is the expected number of solutions for a problem in the ensemble, and N is the number of bits needed to represent a solution (or equivalently the log base 2 of the size of the state space). For instance, for model B, this is:

$$\kappa \quad = \quad \frac{n-1}{2} p_1 \log_m(\frac{1}{1-p_2})$$

This constrainedness parameter, κ lies in the interval $[0, \infty)$. For $\kappa < 1$, problems are under-constrained and are typically easy to show satisfiable. For $\kappa > 1$, problems are over-constrained and are typically relatively easy to show unsatisfiable. For $\kappa \approx 1$, problems are critically constrained and exhibit a sharp transition in satisfiability. For instance, for random constraint satisfaction problem, graph k-colouring problems, number partitioning, and travelling salesperson problems, a rapid phase transition in problem satisfiability has been observed around $\kappa \approx 1$ [27].

Exact theoretical results about the location of the phase transition and of the hardness of random constraint satisfaction problems have been harder to obtain than either empirical results or approximate results using "theories" like that of constrainedness. One exception is work in resolution complexity. Most of the standard backtracking algorithms like forward checking and conflict-directed backjumping explore search trees bounded in size by the size of a corresponding resolution refutation. Resolution complexity results can thus be used to place (lower) bounds on problem hardness. For example, random constraint problems almost surely have an exponential resolution complexity when the constraint tightness is small compared to the domain size [68, 66, 25, 89].

18.1.4 Finite-Size Scaling

The scaling of the phase transition with problem size can be modelled using finite-size scaling methods taken from statistical mechanics [60, 30]. In particular, around some critical value of constrainedness κ_c, problems of all sizes are indistinguishable except for a simple change of scale given by a power law in N. Once rescaled, macroscopic properties like the probability that a problem is satisfiable obey simple equations. For example, the probability of satisfiability can be modelled with the simple equation:

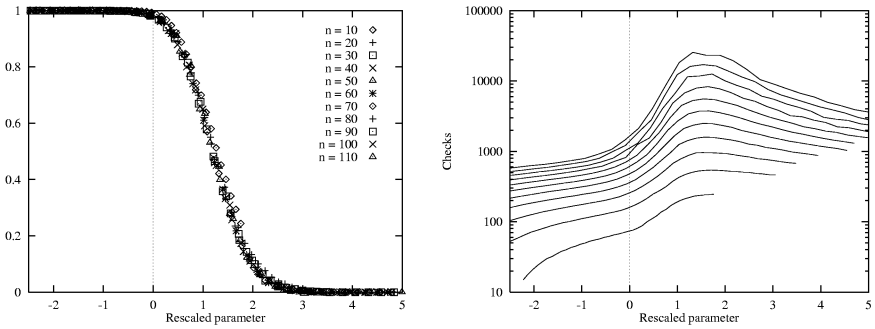
$$prob(Sol > 0) \quad = \quad f(\frac{\kappa - \kappa_c}{\kappa_c} N^{1/\nu})$$

Where f is some universal function, $\frac{\kappa - \kappa_c}{\kappa_c}$ plays the roles of the reduced temperature $\frac{T - T_c}{T_c}$ as it rescales around the critical point, and $N^{1/\nu}$ is a simple power law that describes the scaling with problem size. See Figure 18.2 for some graphs which illustrate this finite-size scaling. Finite-size scaling is used in statistical mechanics to describe systems like Ising magnets with 10^{20} or more atoms (and thus with $2^{10^{20}}$ or so states). It is remarkable therefore that similar mathematics can be used to describe a constraint satisfaction problem with tens or hundreds of variables and therefore just 2^{100} or so states.

Finite-size scaling also appears to be useful to model the change in problem hardness with problem size and problem constrainedness [31]. Finally, parameters like κ and proxies for them which are cheaper to compute appear useful as branching heuristics [27]. A good heuristic is to branch on the "most constrained" variable. This will encourage propagation and tend to give a new subproblem to solve which is much smaller.

18.1.5 Flaws and Flawless Methods

Random problems may contain structures which make them artificially easy. One issue is trivial flaws which a polynomial algorithm could easily discover. In a binary constraint satisfaction problem, the assignment of a value to a variable is said to be flawed if there exists another variable that cannot be assigned a value without violating a constraint. The



(a) Probability of solubility (b) Median Checks

Figure 18.2: Finite-size scaling of the phase transition for Model B problems with $\langle n, 3, p_1, 2/9\rangle$. (a) percentage satisfiability and (b) median search effort for FC-CBJ with the fail-first heuristic against the rescaled parameter, $\frac{\kappa-\kappa_c}{\kappa_c}N^{1/\nu}$ for $\kappa_c = 0.625$ and $\nu = 2.3$. Graphs taken from [27].

value is supported otherwise. A variable is flawed iff each value is flawed. A problem with a flawed variable cannot have a solution. Achlioptas et al. [4] identify a potential shortcoming of all four random models. They prove that if $p_2 \geq 1/m$ then, as n goes to infinity, there almost surely exists a flawed variable. Such problems are not intrinsically hard as a simple arc-consistency algorithm can solve them in polynomial time.

Fortunately, such flaws are unlikely in the size of problems used in practice [27]. We can also define parameters for existing methods and new generation methods which prevent flaws. For example:

model B: the parameter scheme $m = n^\alpha$, $p_1 = \beta \log(n)/(n-1)$ for some constants α, β [91, 89]; Xu and Li also present a similar parameter scheme for model D in which domain size grows polynomially with the number of variables; such problems are guaranteed to have a phase transition and to give problems which almost surely have an exponential resolution complexity;

model D: Smith proposes a somewhat more complex scheme which increases m and the average degree of the constraint graph with n [81];

model E: a new generation method in which we select uniformly, independently and with repetition, exactly $pm^2n(n-1)/2$ nogoods out of the $m^2n(n-1)/2$ possible for some fixed p [4];

modified models A to D: we ensure the conflict matrix of each constraint is flawless by randomly choosing a permutation π_i of 1 to m, and insist that (i, π_i) is a good before we randomly pick nogoods from the other entries in the conflict matrix; each value is thereby guaranteed to have some support [27].

Model E is very similar to the one studied by Williams and Hogg [86]. One possible shortcoming of Model E is that it generates problems with a complete constraint graph for quite small values of p. It is hard therefore to test the performance of algorithms on sparse problems using Model E [27].

The modified versions of models A to D are guaranteed not to contain trivial flaws which would be uncovered by enforcing arc-consistency. However, more recent results have shown that such problems may still be asymptotically unsatisfiable and can be solved in polynomial time using a path consistency algorithm [25]. In response, Gao and Culberson propose a method to generate random problems which are weakly path-consistent, and which almost surely have an exponential resolution complexity.

18.1.6 Related Problems

Phase transition behaviour has also been observed in other problems associated with constraint satisfaction problems. This includes problems in both higher and lower complexity classes. For example, phase transition behaviour has been observed in polynomial problems like establishing the arc-consistency of random constraint satisfaction problems [29]. The probability that the problem can be made arc-consistent goes through a rapid transition, and this is associated with a peak for the complexity of coarse grained arc-consistency algorithms. As a second example, phase transition behaviour has been observed in PSPACE-complete problems like the satisfiability of quantified Boolean formulae. We have to be again carefully of generating flawed problems, but if we do, there is a rapid transition in satisfiability, and this is associated with a complexity peak for many search algorithms [37]. As a third and final example, phase transition behaviour has been observed in PP-complete problems like deciding if a Boolean formulae can be satisfied by at least the square-root of the total number of assignments [8].

18.2 Random Satisfiability

One type of constraint satisfaction problem with a special but very simple structure is propositional satisfiability (SAT). In a SAT problem, variables are only Boolean, and constraints are propositional formulae, typically clauses. Many problems of practical and theoretical importance can be easily mapped into SAT. Random SAT problems have been the subject of extensive research. As a result, some of our deepest understanding has come in this area.

18.2.1 Random k-SAT

There exist a number of different classes of random SAT problem. One such problem class is the "constant probability" model in which each variable is included in a clause with a fixed probability. However, this gives problems which are often easy to solve. Following [67], research has focused on the random k-SAT problem class. A random k-SAT problem in n variables consists of m clauses, each of which contains exactly k Boolean variables drawn uniformly and at random from the set of all possible k-clauses. A rapid transition in satisfiability is observed to occur around a fixed ratio of clauses to variables and this appears to be correlated with a peak in search hardness [67]. Such problems are routinely used to benchmark SAT algorithms.

For random 2-SAT, which is polynomial, the phase transition has been proven to occur at exactly $m/n = 1$ [14, 38]. For random k-SAT for $k \geq 3$, exact results have been harder to obtain. For $k = 3$, the phase transition occurs between $3.42 \leq m/n \leq 4.51$. Experiments suggest that the transition is at $m/n = 4.26$. Asymptotically, the satisfiability

transition is "sharp" (that is, it is a step function) [22]. A very recent result proves that the threshold is at $2^k \log(2) - O(k)$, confirming "approximate" results from statistical mechanics using replica methods [5]. Finite-size scaling methods can again be used to model the sharpening of the phase transition with problem size [60].

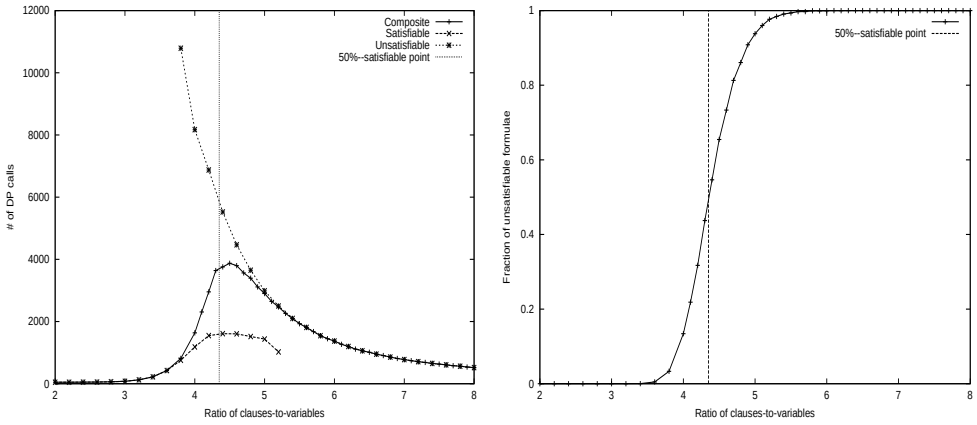


Figure 18.3: Median search cost for DPLL to solve 50 variable random 3-SAT problems and fraction of unsatisfiable clauses, both plotted against the ratio of clauses to variables. Graphs adapted from [67].

At least one note of caution needs to be sounded about the using random 3-SAT as the distribution of solutions is highly skewed. In particular, at the phase transition, the expected number of solutions is exponentially large [57]. Thus, whilst many problems have no solutions, a few problems will have exponentially many.

18.2.2 Backbone

A possible "order parameter" for such phase transitions is the backbone. For a satisfiable problem, the backbone is the fraction of variables which take fixed values in all satisfying assignments. Such variables must be assigned correctly if we are to find a solution. For an unsatisfiable problem, the backbone is the fraction of variables which take fixed values in all assignments which maximize the number of satisfied clauses. A satisfiable problem with a large backbone is likely to be hard to solve for systematic methods like DPLL since there are many variables to branch incorrectly upon. For random 3-SAT, the backbone size jumps discontinuously at the phase transition, suggesting that it behaves like a first-order (or discontinuous) phase transition in statistical mechanics. For random 2-SAT, on the other hand, the backbone size varies smoothly over the phase transition suggesting that it behaves like a second-order (or continuous) phase transition. However, the order (or continuity) of the phase transition does not appear to be directly connected to the problem complexity as there are NP-complete problems with second-order (or continuous) phase transitions.

18.2.3 2+p-SAT

Significant insight into phase transition behaviour has come from "interpolating" between random 2-SAT (which is polynomial and quite well understood theoretically) and random 3-SAT (which is NP-hard and much less well understood theoretically). The random 2+p-SAT problem class consists of SAT problems with a mixture of $(1-p)m$ clauses with 2 variables and pm clauses with 3 variables, each clause drawn uniformly and at random from the space of all possible clauses of the given size. For $p = 0$, we have random 2-SAT. For $p = 1$, we have random 3-SAT. For $0 < p < 1$, we have problems with a mixture of both 2-clauses and 3-clauses. From the perspective of worst-case complexity, 2+p-SAT is rather unexciting. For any fixed $p > 0$, the problem class is NP-complete. However, problems appear to be behave polynomially for $p < 0.4$ [69, 3]. It is only for $p \geq 0.4$ that problems appear hard to solve. This increase in problem hardness has been correlated with a rapid transition in the size of the backbone, and with a change from a continuous to a discontinuous phase transition [69]. For $0 \leq p \leq 0.4$, the satisfiability phase transition for random 2+p-SAT occurs at a simple lower bound, $1/(1-p)$ constructed by simply considering the satisfiability of the embedded 2-SAT subproblem. In other words, the 2-clauses alone determine satisfiability. It is not perhaps so surprising therefore that average search costs appears to be polynomial. Note that, having made some branching decisions on a 3-SAT problem, DPLL is effectively solving a 2+p-SAT subproblem. The performance of such procedures can thus be modelled by mapping trajectories through p and m/n space [15].

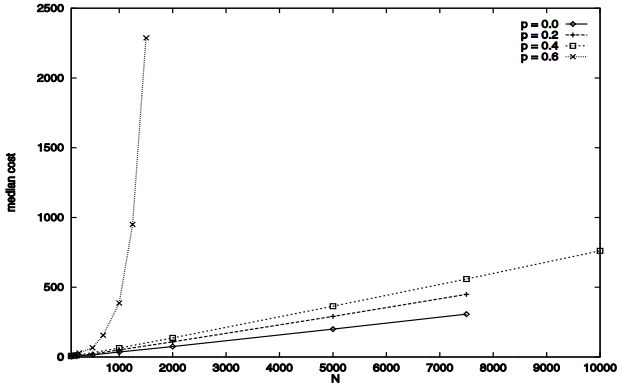


Figure 18.4: Median computational cost for DPLL to solve random 2+p-SAT problems plotted against the number of variables, N for a range of values of p. Graph adapted from [15].

18.2.4 Beyond k-SAT

Phase transition behaviour has been observed in other satisfiability problems including:

1 in k-SAT: each "clause" contains k literals, exactly one of which must be true. This was the first problem NP-complete class in which the exact location of its satisfiability

phase transition was proven [1]. For all $k \geq 3$, random 1 in k-SAT problems have a sharp, second-order or continuous phase transition at $m/n = 2/k(k-1)$.

NAE-SAT: each "clause" contains k literals, all of which cannot take the same truth value. For $k = 3$, the phase transition for random NAE SAT problems occurs somewhere between $1.514 < m/n < 2.215$ [1]. Empirical results put the phase transition at around $m/n \approx 2.1$. A NAE SAT problem can be mapped into a SAT problem with twice the number of clauses. Although these clauses are correlated, it is remarkable that these correlations appear to be largely irrelevant and the phase transition occurs at almost exactly half the clause to variable ratio of the random 3-SAT phase transition.

XOR SAT: each "clause" contains k literals, an odd number of which must be true in any satisfying assignment. For $k = 3$, random NAE SAT has a sharp threshold in the interval $0.8894 \leq m/n \leq 0.9278$ [17]. Experiments put the transition at $m/n \approx 0.92$, whilst statistical mechanical calculations put it at $m/n = 0.918$ [21].

non-clausal SAT: formulae have a fixed shape (a given structure of and and or connectives) which are labelled with literals at random [71]. This model displays a phase transition in satisfiability with an associated easy-hard-easy pattern in search cost.

quantified SAT: in a quantified Boolean formula (QBF) we have variables which are both existentially quantified and universally quantified. If we generate random QBF formulae, we need to throw out clauses containing just universally quantified variables (as these are trivially unsatisfiable). If we eliminate such "flaws", there is a rapid phase transition, and an associated complexity peak [37]

18.2.5 Satisfiable Problems

Random problems have been a driving force in the design of better algorithms. To benchmark incomplete local search procedures, standard random problem generators are unsuitable as they produce both satisfiable and unsatisfiable instances. We could simply filter out unsatisfiable instances using a complete method. However, we are then unable to benchmark incomplete search methods on problems that are beyond the reach of complete methods. Designing generators, on the other hand, that generate only satisfiable problems has proven surprisingly difficult.

One approach is to "hide" at least one solution in a problem instance. For example, we can choose a random truth assignment $T \in \{0,1\}^n$ and then generate a formula with n variables and αm random clauses, rejecting any clause that violates T. Unfortunately, this method is highly biased to generating formulas with many assignments. They are much easier for local search methods like Walksat [74] than formulas of comparable size obtained by filtering a random 3-SAT generator. More sophisticated versions of this "1-hidden assignment" scheme provide improvements but still lead to biased samples [7]. Achlioptas et al. [6] proposed a "2-hidden assignment" approach in which clauses that violate both T and its complement are rejected. Whilst DPLL solvers find such problems as hard as regular random 3-SAT problems, local search methods find them easy. An improved approach, called "q-hidden" [56], hides a single assignment but biases the distribution so that each variable is as likely to appear positively as as negatively, and the formula no longer

points toward the satisfying assignment T. Indeed, we can even make it more likely that a variable occurrence disagrees with T, so that the formula becomes "deceptive" and points away from the hidden assignment. Empirical results suggest that the q-hidden model produces formulas that are much harder for Walksat.

Recently Xu et al [90] gave modifications of the random models B and D to generate "forced" solvable instances whose hardness is comparable to "unforced" solvable instances, based on the theoretical argument that the number of expected solutions in both cases is identical. They also provide empirical results showing that the unforced solvable instances, unforced solvable and unsolvable instances, and forced solvable instances exhibit a similar hardness pattern. In section 18.3 we will discuss a quite different strategy for generating guaranteed satisfiable random instances for structured CSP problems.

18.2.6 Optimization Problems

Phase transition behaviour has also been identified in a range of optimization problems. Some of our best understanding has come in satisfiability problems related to optimization like MAX-SAT (for example, [94, 79]. However, insight has also come from other domains like number partitioning [34, 36] and the symmetric and asymmetric travelling salesperson problems [35, 96, 95]. The simplest view is that optimization problems naturally push us to the phase boundary [33]. For systematic backtracking algorithms like branch and bound, we essentially solve a pair of decision problems right at the phase transition: we first find a solution to the decision problem with an optimal objective and then prove that the decision problem with any smaller objective is unsatisfiable. A more sophisticated view is that optimization problems like MAX-SAT can be viewed as bounded by a sequence of decision problems at successive objective values [94].

The concept of backbone has also been generalized to deal with optimization problems [78]. As with decision problems, transitions in problem hardness have been correlated with rapid transitions in backbone size [78, 94, 95]. These views suggest that there is a relatively simple connection between the hardness of decision and the corresponding optimization problem. Indeed, by solving (easy) decision problems away from the phase boundary, we can often predict the cost of finding optimal solutions [77].

18.3 Random Problems with Structure

Uniform random problems like random k-SAT are unlikely to contain structures found in many real world problems. Such structures can make problems much easier or much harder to solve. Researchers have therefore looked at ways of generating structured random problems. For example, the question of the existence of discrete structures like quasigroups with particular properties gives some of the most challenging search problems [76]. However, such problems may be too uniform and highly structured when compared to messy real-world problems. In order to bridge this gap, a number of random problem classes have been proposed that incorporate structures rarely seen in purely uniform random problems. For example, Gomes and Selman [39] proposed the quasigroup completion problem (QCP). As another example, Walsh proposed small-world search problems [85].

18.3.1 Quasigroup Completion

An order n quasigroup, or *Latin Square* , is defined by $n \times n$ multiplication in which each row and column is a permutation of the n symbols. A *partial* Latin square with p pre-assigned cells is an $n \times n$ matrix in which p cells of the matrix have been assigned symbols such that no symbol occurs repeated in a row or a column. The Quasigroup Completion Problem (QCP) is to determine if the remaining $n^2 - p$ cells (or "holes") can be assigned to obtain a complete Latin square (see Figure18.5). QCP is NP-complete [16]. The structure in QCP is similar to that found in real-world domains like scheduling, timetabling, routing, and experimental design. One problem that directly maps onto the QCP is that of assigning wavelengths to routes in fiber-optic networks [61].

	1	2	3
2		4	1
1	4		2
3		1	

4	1	2	3
2	3	4	1
1	4	3	2
3	2	1	4

Figure 18.5: Quasigroup Completion Problem of order 4, with 5 holes.

To generate a random QCP instance, we randomly select p cells and assign each a symbol. We have a choice in the level of consistency enforced between such assignments to eliminate "obvious" inconsistencies. The most commonly used model enforces forward checking [39]. Shaw et al [75] studied a model which enforces generalized arc consistency on the all-different constraints on the rows and the columns of the matrix. This gives harder problems but biases the sampling. Empirical studies have identified phase transition behaviour in QCP [39]. The computationally hardest instances again lie at the phase transition. almost all unsolvable ("over-constrained" region). Figure 18.6 shows the computational cost (median number of backtracks) and phase transition in solvability for solving QCP instances of different orders.

18.3.2 Quasigroup with Holes

The QCP model generates both satisfiable and unsatisfiable instances. A different model, the Quasigroup With Holes problem (QWH), generates *only* satisfiable instances with good computational properties [2]. QWH instances are generated by starting with a full quasigroup and "punching" holes into it. Achlioptas et al [2] proposed the following QWH generator: (1) Generate a complete Latin square uniformly from the space of all Latin squares using a Markov chain; (2) punch a fraction p of "holes" into the full Latin square (*i.e.*, unassign some of the entries) in a uniform way. The resulting partial Latin square is guaranteed to be satisfiable. Achlioptas et al [2] demonstrated a rapid transition in the size of the backbone of QWH instances, coinciding with the hardest problem instances for *both* incomplete and complete search methods. Note that this transition is different from the standard transition in satisfiability as QWH *only* contains satisfiable instances. The location of this transition appears to scale as $n^2 - p/n^{1.55}$ [2].

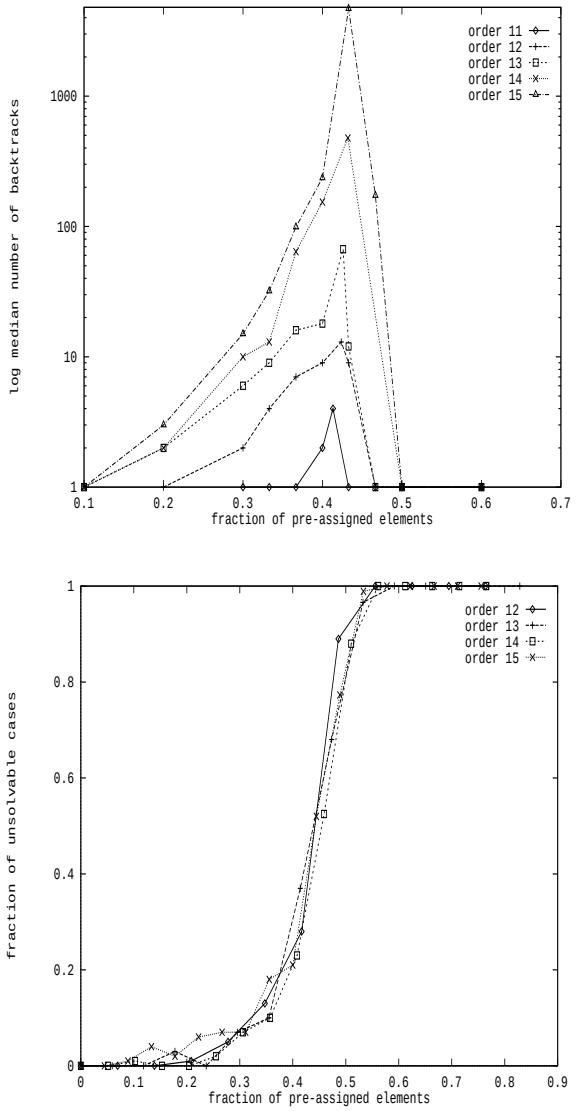


Figure 18.6: Top panel: computational cost of solving QCP instances (order 11–15). X-axis: fraction of pre-assigned cells; Y-axis - median number of backtracks for solution (log scale). Bottom panel: phase transition in solvability for QCP instances (order 12–15). X-axis: fraction of pre-assigned cells; Y-axis - fraction of instances for which the partial Latin square could not be completed into a full Latin square. (Each data point was computed based on 100 instances. Graphs from [39].)

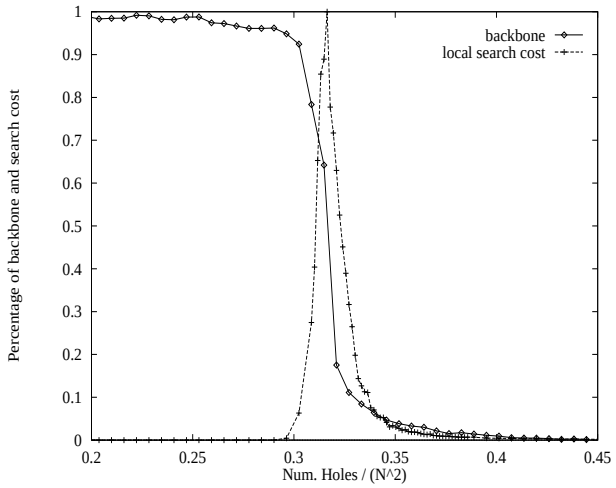


Figure 18.7: Backbone phase transition with cost profile for QWH. Graph from [2].

18.3.3 Other Structured Problems

A number of other random problem classes with structure have been studied. For instance, Walsh looked at search problems like graph coloring where the underlying graph has a "small-world" structure [85]. Although small-world graphs are sparse, their nodes tend to be clustered and the path length between any two nodes short. Walsh showed that a small-world structure often occurs in graphs associated with many real-world search problems. Unfortunately the cost of coloring random graphs with a small-world structure can have a heavy-tailed distribution (see next section) in which a few runs are exceptionally long. However, the strategy of randomization and restarts can eliminate these heavy tails.

To generate random small-world graphs, Walsh merged together random graphs with a structured ring lattice [85]. Inspired by this method, Gent *et al.* proposed a general method called *morphing* to introduce structure or randomness into a wide variety of problems [28]. They show that a mixture of structure and randomness can often make search problems very hard to solve. A little structure added to a random problem, or a little randomness added to a structured problem may be enough to mislead search heuristics. They argue that morphing provides many of the advantages of random and structured problem classes without some of the disadvantages. As in random problem classes, we can generate large, and statistically significant samples with ease. However, unlike random problems, morphed problems can contain many of the structures met in practice.

18.4 Runtime Variability

Broadly speaking, random problems tend to display "easy-hard-easy" patterns in difficulty. However, there has been some research into variability within this simple picture, and into ways such variability can be exploited.

18.4.1 Randomization

A randomized complete algorithm can be viewed as a probability distribution on a set of deterministic algorithms. Behaviour can vary even on a single input, depending on the random choices made by the algorithm. The classical adversary argument for establishing lower bounds on the run-time of a deterministic algorithm is based on the construction of a input on which the algorithm performs poorly. While an adversary may be able to construct an input that foils one (or a small fraction) of the deterministic algorithms in the set, it is more difficult to devise inputs that are likely to defeat a randomly chosen algorithm. Furthermore, as we will discuss below, the introduction of a "small" random element allows one to run the randomized method on the *same* instance several times, isolating the variance inherent in the search procedure from e.g., the variance that would result from considering different instances.

There are several opportunities to introduce randomization in a backtrack search method. For example, we can add randomization to the branching heuristic for tie-breaking [41, 43]. Even this simple modification can dramatically change the behavior of a search algorithm. If the branching heuristic is particular decisive, it may rarely need to tie-break. In this case, we can tie-break between some of the top ranked choices. The look-ahead and look-back procedures can also be randomized. Lynce *et al.* random backtracking which randomizes the backtracking points, and unrestricted backtracking which combines learning to maintain completeness [64, 65]. Another example is restarts of a deterministic backtrack solver with clause learning: each time the solver is restarted, with the additional learned clauses, it behaves quite differently from the previous run, appearing to behave "randomly" [70, 65].

18.4.2 Fat and Heavy Tailed Behavior

The study of the runtime distributions instead of just medians and means often provides a better characterization of search methods and much useful information in the design of algorithms. For instance, complete backtrack search methods exhibit *fat* and *heavy-tailed* behavior [47, 41, 23]. *Fat-tailedness* is based on the *kurtosis* of a distribution. This is defined as μ_4/μ_2^2 where μ_4 is the fourth central moment about the mean and μ_2 is the second central moment about the mean, *i.e.*, the variance. If a distribution has a high central peak and long tails, than the kurtosis is large. The kurtosis of the standard normal distribution is 3. A distribution with a kurtosis larger than 3 is *fat-tailed* or *leptokurtic*. Examples of distributions that are characterized by fat-tails are the exponential distribution, the lognormal distribution, and the Weibull distribution. Heavy-tailed distributions have "heaver" tails than fat-tailed distributions; in fact they have some infinite moments. More precisely, a random variable X is heavy-tailed if it has Pareto like decay in its distribution, i.e:

$$1 - F(x) = P[X > x] \sim Cx^{-\alpha},\ x > 0,$$

where $\alpha > 0$ and $C > 0$ are constants. When $1 < \alpha < 2$, X has infinite variance, and infinite mean and variance when $0 < \alpha <= 1$. The log-log plot of $1 - F(x)$ of a Pareto-like distribution (i.e., the survival function) shows linear behavior with slope determined by α.

Backtrack search methods exhibit dramatically different statistical regimes across the constrainedness regions of random CSP models [11]. Figure 18.8 illustrates the phenomenon. In the first regime (the bottom two curves in figure 18.8, $p \leq 0.07$), we see

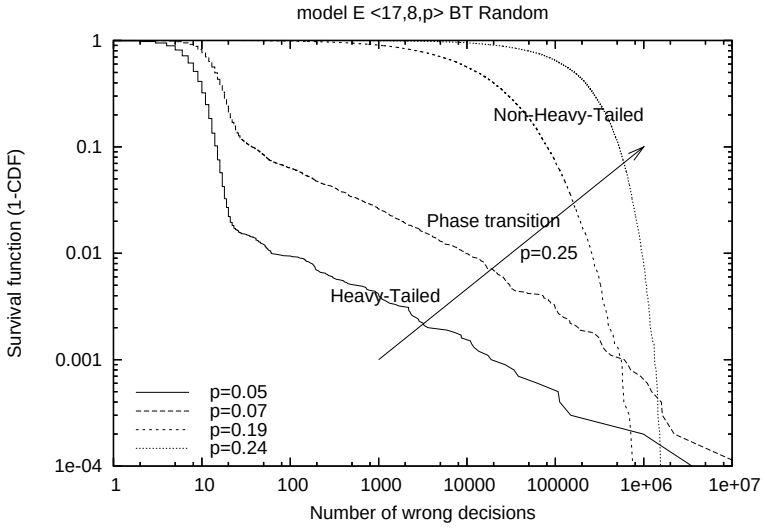


Figure 18.8: Heavy-tailed (linear behavior) and non-heavy-tailed regime in the runtime of instances of model E $\langle 17, 8, p\rangle$. CDF stands for Cumulative Density Function. Graphs adapted from [11].

heavy-tailed behavior. This means that the runtime distributions decay slowly. When we increase the constrainedness of our model towards the phase transition (higher p), we encounter a different statistical regime in the runtime distributions, where the heavy-tails disappear. In this region, the instances become inherently hard for the backtrack search algorithm, all the runs become homogeneously long, the variance of the backtrack search algorithm decreases and the tails of its survival function decay rapidly (see top two curves in figure 18.8, with $p = 0.19$ and $p = 0.24$; tails decay exponentially).

Heavy-tailed behavior in combinatorial search has been observed in several other domains, in both random instances and real-world instances: QCP [41], scheduling [45], planning[44], graph coloring [85, 55], and inductive logic programming [92]. Several formal models generating heavy-tailed behavior in search have been proposed [13, 87, 88, 55, 11, 51]. If a runtime distribution of a backtrack search method is heavy-tailed, it will produce runs over several orders of magnitude, some extremely long but also some extremely short. Methods like randomization and restarts try to exploit this phenomenon. (See section 18.4.4.)

18.4.3 Backdoors

Insight into heavy-tailed behaviour comes from considering backdoor variables. These are variables which, when set, give us a polynomial subproblem. Intuitively, a small backdoor set explains how a backtrack search method can get "lucky" on certain runs, where backdoor variables are identified early on in the search and set the right way. Formally, the definition of a backdoor depends on a particular algorithm, referred to as *sub-solver*, that

solves a tractable subcase of the general constraint satisfaction problem [87].

Definition 18.1. *A* sub-solver A *given as input a CSP,* C*, satisfies the following:*

• *(Trichotomy)* A *either rejects the input* C*, or "determines"* C *correctly (as unsatisfiable or satisfiable, returning a solution if satisfiable),*

• *(Efficiency)* A *runs in polynomial time,*

• *(Trivial solvability)* A *can determine if* C *is trivially true (has no constraints) or trivially false (has a contradictory constraint),*

• *(Self-reducibility) if* A *determines* C*, then for any variable* x *and value* v*, then* A *determines* $C[v/x]$.[1]

For instance, A could be an algorithm that performs unit propagation, or arc consistency, or hyper-arc consistency for the *alldiff* constraint, or an algorithm that solves a linear programming problem, or any algorithm satisfying the above four properties. Using the definition of sub-solver we can now formally define the concept of backdoor set. Let A be a sub-solver, and C be a CSP. A nonempty subset S of the variables is a *backdoor* in C for A if for some $a_S : S \to D$, A returns a satisfying assignment of $C[a_S]$. Intuitively, the backdoor corresponds to a set of variables, such that when set correctly, the sub-solver can solve the remaining problem. A stronger notion of the backdoor, considers both satisfiable and unsatisfiable (inconsistent) problem instances. A nonempty subset S of the variables is a *strong backdoor* in C for A if for all $a_S : S \to D$, A returns a satisfying assignment or concludes unsatisfiability of $C[a_S]$. From a logical perspective, there is no formal connection between the backbone and the backdoor of a problem. Indeed, whilst it is possible to exhibit problems where they are identical, it is also possible to exhibit problems where they are disjoint. In practice, the overlap between backbones and backdoors appears to be slight [59].

Cutsets [18] are a particular kind of backdoor sets. A cutset is a set of variables such that, once they are removed from the constraint graph, the remaining graph has a property that enables efficient reasoning, an *induced width* of at most a constant bound b; for example, if $b = 1$ then the graph is cycle-free, i.e., it can be viewed as a tree, and therefore it can be solved using directed arc consistency. Backdoor sets can thus be seen as a generalization of cutsets, i.e., any cutset is a backdoor set. Backdoors are more general than the notion of cutsets since they consider *any* kind of polynomial time sub-solver. Note that, while cutsets (and W-cutsets) use a notion of tractability based solely on the topology of the underlying constraint graph, backdoor sets rely on a polynomial time solver to define the notion of tractability. A related issue is the fact that backdoor sets factor in the values of variables and the semantics of constraints (via the propagation triggered by the polytime solver) and therefore backdoor sets can be significantly smaller than cutsets. For example, if we have a constraint graph that contains a clique of size k, the cutset has at least $k - 2$ variables, while the backdoor set can be substantially smaller. Another example, considering CNF theories, is that while a Horn theory can have a cutset of size $O(n)$, the backdoor with respect to unit propagation has size 0 - unit propagation immediately detects (in)consistency of Horn theories. Stated differently, given two CNF theories, one of them a Horn theory and the other one an arbitrary CNF theory but with the same constraint graph as the Horn theory, there is no difference between the two theories from the perspective of

[1] *We use the notation* $C[v/x]$ *to denote the simplifi ed CSP obtained from a CSP,* C*, by setting the value of variable* x *to value* v.

cutsets, but the difference between them from the perspective of backdoors is likely to be substantial.

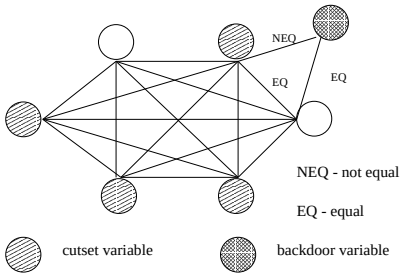
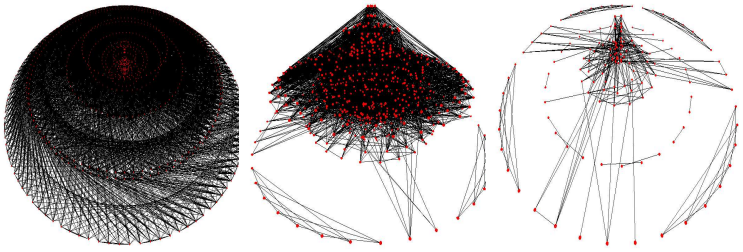


Figure 18.9: Cutset vs. backdoor sets. Any cutset is a backdoor set. However, backdoor sets can be considerably smaller since they factor in the semantics of the constraints, via the propagation triggered by the sub-solver. Any clique of size k has a cutset of size $k-2$. In this picture, the size of the cutset is 4 while the size of the backdoor set is 1 if the sub-solver performs forward checking or anything stronger.

A key issue is therefore the size of the backdoor set. Random formulas do not appear to have small backdoor sets. For example, for random 3-SAT problems, the backdoor set appears to be a constant fraction (roughly 30%) of the total number of variables [53]. This may explain why the current DPLL based solvers have not made significant progress on hard randomly generated instances. Seizer considers the parameterized complexity of the problem of whether a SAT instance has a weak or strong backdoor set of size k or less for DPLL style sub-solvers, i.e., subsolvers based on unit propagation and/or pure literal elimination [82]. He shows that detection of weak and strong backdoor sets is unlikely to be fixed-parameter tractable. Nishimura et al. [72] provide more positive results for detecting backdoor sets where the sub-solver solves Horn or 2-cnf formulas, both of which are linear time problems. They prove that the detection of such a strong backdoor set is fixed-parameter tractable, whilst the detection of a weak backdoor set is not. The explanation that they offer for such a discrepancy is quite interesting: for strong backdoor sets one only has to guarantee that the chosen set of variables gives a subproblem with the chosen syntactic class; for weak backdoor sets, one also has to guarantee satisfiability of the simplified formula, a property that cannot be described syntactically.

Empirical results based on real-world instances suggest a more positive picture. Structured problem instances can have surprisingly small sets of backdoor variables, which may explain why current state of the art solvers are able to solve very large real-world instances. For example the logistics-d planning problem instance, (log.d) has a backdoor set of just 12 variables, compared to a total of nearly 7,000 variables in the formula, using the polytime propagation techniques of the SAT solver, Satz [62]. Hoffmann et al proved the existence of *strong* backdoor sets of size just $O(\log(n)$ for certain families of logistics planning problems and blocks world problems domains [54].

Even though, computing backdoor sets is typically intractable, even if we bound the size of the backdoor [82], heuristics and techniques like randomization and restarts may nevertheless be able to uncover a small backdoor in practice [87, 59, 52]. For example one can obtain a complete randomized restart strategy that runs in polynomial time when



(a) (b) (c)

Figure 18.10: Constraint graph of a real-world instance from the logistics planning domain. The instance in the plot has 843 vars and 7,301 clauses. One backdoor set for this instance w.r.t. unit propagation has size 16 (not necessarily the minimum backdoor set). (a) Constraint graph of the original constraint graph of the instance. (b) Constraint graph after setting 5 variables and performing unit propagation on the graph. (c) Constraint graph after setting 14 variables and performing unit propagation on the graph.

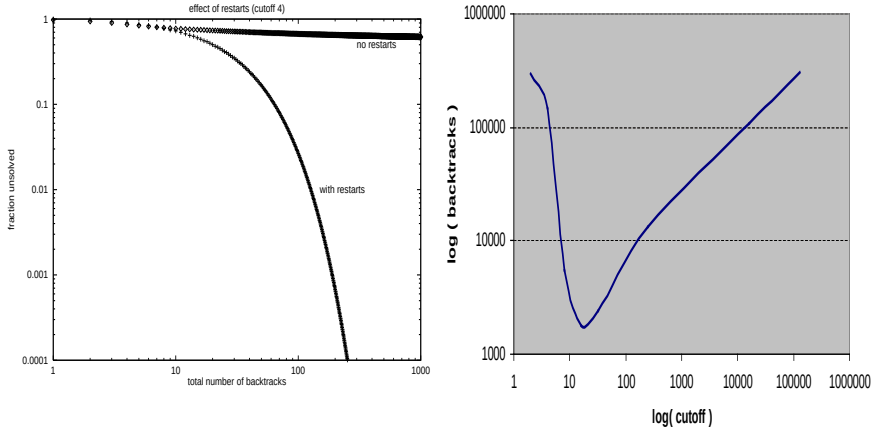
the backdoor set contains at most $\log(n)$ variables [87]. Dequen and Dubois introduced a heuristic for DPLL based solvers that exploits the notion of backbone that outperforms other heuristics on random 3-SAT problems [19, 20].

18.4.4 Restarts

One way to exploit heavy-tailed behaviour is to add restarts to a backtracking procedure. A sequence of short runs instead of a single long run may be a more effective use of computational resources. Gomes et al. proposed a rapid randomization and restart (RRR) to take advantage of heavy-tailed behaviour and boost the efficiency of complete backtrack search procedures [44]. In practice, one gradually increases the cutoff to maintain completeness ([44]). Gomes et al. have proved formally that a restart strategy with a fix cutoff eliminates heavy-tail behavior and therefore all the moments of a restart strategy are finite [43].

When the underlying runtime distribution of the randomized procedure is fully known, the optimal restart policy is a fixed cutoff [63]. When there is no *a priori* knowledge about the distribution, Luby *et al.* also provide a *universal strategy* which minimizes the expected cost. This consists of runs whose lengths are powers of two, and each time a pair of runs of a given length has been completed, a run of twice that length is immediately executed. The universal strategy is of the form: $1, 1, 2, 1, 1, 2, 4, 1, 1, 2, 4, 8, \cdots$. Although the universal strategy of Luby *et al.* is provably within a constant log factor of the the optimal fixed cutoff, the schedule often converges too slowly in practice. Walsh introduced a restart strategy, inspired by Luby *et al.*'s analysis, in which the cutoff value increases geometrically [85]. The advantage of such a strategy is that it is less sensitive to the details of the underlying distribution. State-of-the-art SAT solvers now routinely use restarts. In practice, the solvers use a default cutoff value, which is increased, linearly, every given number of restarts, guaranteeing the completeness of the solver in the limit ([70]). Another important feature is that they learn clauses across restarts. The work on backdoor sets also provides formal results on restart strategies. In particular, even though finding a small set of backdoor variables is computationally hard, the presence of a small backdoor in a

problem provides a concrete computational advantage in solving it with restarts [87].



(a) (b)

Figure 18.11: Restarts: (a) Tail $(1 - F(x))$ as a function of the total number of backtracks for a QCP instance, log-log scale; the left curve is for a cutoff value of 4; and, the right curve is without restarts. (b) The effect of different cutoff values on solution cost for the logistics.d planning problem. Graph adapted from [41, 43].

In reality, we will be somewhere between full and no knowledge of the runtime distribution. [48] introduce a Bayesian framework for learning predictive models of randomized backtrack solvers based on this situation. Extending that work, [58], considered restart policies that can factor in information based on real-time observations about a solver's behavior. In particular, they introduce an *optimal* policy for dynamic restarts that considers observations about solver behavior. They also consider the dependency between runs. They give a dynamic programming approach to generate the optimal restart strategy, and combine the resulting policy with real-time observations to boost performance of backtrack search methods.

Variants of restart strategies include randomized backtracking [64], and the random jump strategy [93] which has been used to solve a dozen previously open problems in finite algebra. Finally, one can also take advantage of the high variance of combinatorial search methods by combining several algorithms into a "portfolio," and running them in parallel or interleaving them on a single processor [50, 40, 42].

18.5 History

Research in this area can be traced back at least as far as Erdös and Rényi's work on phase transition behaviour in random graphs [10]. One of the first observations of a complexity peak for constraint satisfaction problems was Gaschnig in his PhD thesis where he used $\langle 10, 10, 1, p_2 \rangle$ model B problems (these resemble 10-queens problems) [26]. Fu and Anderson connected phase transition behaviour with computational complexity [24], as did Huberman and Hogg [49]. However, it was not till 1991, when Cheeseman, Kanefsky and Taylor published an influential paper [12] that research in this area accelerated rapidly.

Cheeseman et al. correlated complexity peaks for search algorithms with rapid transitions in problem satisfiability. They conjectured that all NP-complete problems display such phase transition behaviour and that this is correlated with the rapid change in solution probability. More recently, phase transition behaviour has been correlated with rapid changes in the size of the backbone. However, problem classes have been identified like Hamiltonian Cycle whose phase transition does not seem to throw up hard instances [83], as well as NP-complete problem classes which do not have any backbone [9]. Cheeseman, Kanefsky and Taylor also conjectured that polynomial problems do not have such phase transition behaviour or if they do it occurs only for a bounded problem size (and hence bounded cost) [12]. However, as we noted, even polynomial problems like establishing arc-consistency display similar phase transition behaviour [29]. Another polynomial problem class which displays phase transition behaviour is 2-SAT [38, 14].

18.6 Conclusions

As the many examples in this chapter have demonstrated, research into random problems has played a significant role in our understanding of problem hardness, and in the design of efficient and effective algorithms to solve constraint satisfaction and optimization problems. We need to take care when using random problems as there are a number of pitfalls awaiting the unwary. For example, random problems may lack structures found in real world problems. Research into areas like random quasigroup completion attempts to address such issues directly. As a second example, random problems may be generated with "flaws". However, if care is taken, such flaws can easily be prevented. There are many areas that look promising for future research. For example, we are only starting to understand the connection (if any) between the backbone and backdoor [59]. As another example, random problems capturing structural properties of real world problems [54] are starting to provide insight into key issues like backdoors. As a final example, search methods inspired by insights from random problems like randomization and restarts offer a promising new way to tackle hard computational problems. What is certain, however, is that random problems will continue to be a useful tool in understanding (and thus tackling) problem hardness.

Bibliography

[1] D. Achlioptas, A. Chtcherba, G. Istrate, and C. Moore. The phase transition in 1-in-k SAT and NAE SAT. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'01)*, pages 719–720, 2001.

[2] D. Achlioptas, C. Gomes, H. Kautz, and B. Selman. Generating Satisfiable Instances. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-00)*, New Providence, RI, 2000. AAAI Press.

[3] D. Achlioptas, L.M. Kirousis, E. Kranakis, and D. Krizanc. Rigorous results for (2+p)-SAT. *Theoretical Computer Science*, 265(1-2):109–129, 2001.

[4] D. Achlioptas, L.M. Kirousis, E. Kranakis, D. Krizanc, M.S.O. Molloy, and Y.C. Stamatiou. Random constraint satisfaction: A more accurate picture. In G. Smolka, editor, *Proceedings of Third International Conference on Principles and Practice of Constraint Programming (CP97)*, pages 107–120. Springer, 1997.

[5] D. Achlioptas and Y. Peres. The threshold for random k-SAT is $2^k \log(2) - o(k)$. *Journal of the AMS*, 17(4):947–973, 2004.

[6] D. Achlioptas, H. Jia, and C. Moore. Hiding satisfying assignments: Two are better than one. In *Proceedings of AAAI 2004*. AAAI, 2004.

[7] Y. Asahiro, K. Iwama, and E. Miyano. Random generation of test instances with controlled attributes. Contributed to the DIMACS 1993 Challenge archive, 1993.

[8] D.D. Bailey, V. Dalmau, and P.G. Kolaitis. Phase transitions of PP-complete satisfiability problems. In *Proceedings of the 17th IJCAI*, pages 183–189. International Joint Conference on Artificial Intelligence, 2001.

[9] A.J. Beacham. The complexity of problems without backbones. Master's thesis, Department of Computing Science, University of Alberta, 2000.

[10] B. Bollobás. *Random Graphs*. London, Academic Press, 1985.

[11] C. Gomes and C. Fernandez and B. Selman and C. Bessiere. Statistical Regimes Across Constrainedness Regions. In M. Wallace, editor, *Proceedings of 10th International Conference on Principles and Practice of Constraint Programming (CP2004)*. Springer, 2004.

[12] P. Cheeseman, B. Kanefsky, and W.M. Taylor. Where the really hard problems are. In *Proceedings of the 12th IJCAI*, pages 331–337. International Joint Conference on Artificial Intelligence, 1991.

[13] H. Chen, C. Gomes, and B. Selman. Formal Models of Heavy-tailed Behavior in Combinatorial Search. In T. Walsh, editor, *Proceedings of 7th International Conference on Principles and Practice of Constraint Programming (CP2001)*, pages 408–421. Springer, 2001.

[14] V. Chvatal and B. Reed. Mick gets some (the odds are on his side). In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 620–627. IEEE, 1992.

[15] S. Cocco and R. Monasson. Trajectories in phase diagrams, growth processes and computational complexity: how search algorithms solve the 3-satisfiability problem. *Physical Review Letters*, 86(8):1654–1657, 2001.

[16] C. Colbourn. The complexity of completing partial Latin squares. *Discrete Applied Mathematics*, 8:25–30, 1984.

[17] N. Creognou, H. Daude, and O. Dubois. Approximating the satisfiability threshold for random k-XOR-formulas. Technical Report LATP/UMR6632 01-17, Laboratoire d'Informatique Fondamentale de Marseille, 2001.

[18] R. Dechter. Enhancement schemes for constraint processing: Backjumping, learning and cutset decomposition. *Artificial Intelligence*, 41(3):273–312, 1990.

[19] G. Dequen and O. Dubois. Kcnfs: An efficient solver for random k-SAT formulae. In *Proceedings of Sixth International Conference on Theory and Applications of Satisfiability Testing (SAT-03)*, 2003.

[20] O. Dubois and G. Dequen. A backbone search heuristic for efficient solving of hard 3-SAT formulae. In *Proceedings of the 18th IJCAI*. International Joint Conference on Artificial Intelligence, 2003.

[21] S. Franz, M. Leone, F. Ricci-Tersenghi, and R. Zecchina. Exact solutions for diluted spin glasses and optimization problems. *Phys. Rev. Letters*, 87(12):127209, 2001.

[22] E. Friedgut and J. Bourgain. Sharp thresholds of graph properties and the k-SAT problem. *Journal of the American Mathematical Society*, 12(4):1017–1054, 1999.

[23] D. Frost, I. Rish, and L. Vila. Summarizing CSP Hardness with Continuous Prob-

ability Distributions. In *Proceedings of the 14th National Conference on AI*, pages 327–333. American Association for Artificial Intelligence, 1997.

[24] Y. Fu and P. Anderson. Application of statistical mechanics to NP-complete problems in combinatorial optimisation. *J. Phys. A*, 19:1605–1620, 1986.

[25] Y. Gao and J. Culberson. Consistency and random constraint satisfaction problems. In M. Wallace, editor, *Proceedings of 10th International Conference on Principles and Practice of Constraint Programming (CP2004)*. Springer, 2004.

[26] J. Gaschnig. Performance measurement and analysis of certain search algorithms. Technical report CMU-CS-79-124, Carnegie-Mellon University, 1979. PhD thesis.

[27] 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.

[28] I.P. Gent, H. Hoos, P. Prosser, and T. Walsh. Morphing: Combining structure and randomness. In *Proceedings of the 16th National Conference on AI*. American Association for Artificial Intelligence, 1999.

[29] I.P. Gent, E. MacIntyre, P. Prosser, P. Shaw, and T. Walsh. The constrainedness of arc consistency. In *3rd International Conference on Principles and Practices of Constraint Programming (CP-97)*, pages 327–340. Springer, 1997.

[30] I.P. Gent, E. MacIntyre, P. Prosser, and T. Walsh. The constrainedness of search. In *Proceedings of the 13th National Conference on AI*, pages 246–252. American Association for Artificial Intelligence, 1996.

[31] I.P. Gent, E. MacIntyre, P. Prosser, and T. Walsh. The scaling of search cost. In *Proceedings of the 14th National Conference on AI*, pages 315–320. American Association for Artificial Intelligence, 1997.

[32] I.P. Gent and T. Walsh. Easy Problems are Sometimes Hard. *Artificial Intelligence*, 70:335–345, 1994.

[33] I.P. Gent and T. Walsh. Phase transitions from real computational problems. In *Proceedings of the 8th International Symposium on Artificial Intelligence*, pages 356–364, 1995. URL `http://apes.cs.strath.ac.uk/papers/ISAI95crc.ps.gz`.

[34] I.P. Gent and T. Walsh. Phase transitions and annealed theories: Number partitioning as a case study. In *Proceedings of 12th ECAI*, 1996.

[35] I.P. Gent and T. Walsh. The TSP phase transition. *Artificial Intelligence*, 88:349–358, 1996.

[36] I.P. Gent and T. Walsh. Analysis of heuristics for number partitioning. *Computational Intelligence*, 14(3):430–451, 1998.

[37] I.P. Gent and T. Walsh. Beyond NP: the QSAT phase transition. In *Proceedings of the 16th National Conference on AI*. American Association for Artificial Intelligence, 1999.

[38] A. Goerdt. A threshold for unsatisfiability. In I. Havel and V. Koubek, editors, *Mathematical Foundations of Computer Science*, Lecture Notes in Computer Science, pages 264–274. Springer Verlag, 1992.

[39] C. Gomes and B. Selman. Problem Structure in the Presence of Perturbations. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, pages 221–227, New Providence, RI, 1997. AAAI Press.

[40] C. P. Gomes and B. Selman. Algorithm Portfolio Design: Theory vs. Practice. In *Proceedings of the Thirteenth Conference On Uncertainty in Artificial Intelligence (UAI-97)*, Linz, Austria., 1997. Morgan Kaufman.

[41] C. Gomes, B. Selman, and N. Crato. Heavy-tailed Distributions in Combinatorial Search. In G. Smolka, editor, *Proceedings of Third International Conference on Principles and Practice of Constraint Programming (CP97)*, pages 121–135. Springer, 1997.

[42] C. Gomes and B. Selman. Algorithm portfolios. *Artificial Intelligence*, 126:43–62, 2001.

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

[44] C. Gomes, B. Selman, and H. Kautz. Boosting Combinatorial Search Through Randomization. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98)*. American Association for Artificial Intelligence, 1998.

[45] C. Gomes, B. Selman, K. McAloon, and C. Tretkoff. Randomization in backtrack search: Exploiting heavy-tailed profiles for solving hard scheduling problems. In *The Fourth International Conference on Artificial Intelligence Planning Systems (AIPS'98)*, 1998.

[46] S. Grant and B.M. Smith. Where the *Exceptionally* Hard Problems Are. In *Proceedings of the CP-95 workshop on Really Hard Problems*, 1995. Available as University of Leeds, School of Computer Studies Research Report 95.35.

[47] T. Hogg and C.P. Williams. The Hardest Constraint Problems: a Double Phase Transition. *Artificial Intelligence*, 69:359–377, 1994.

[48] E. Horvitz, Y. Ruan, C. Gomes, H. Kautz, B. Selman, and D. Chickering. A bayesian approach to tackling hard computational problems. In *Proceedings of 17th Annual Conference on Uncertainty in Artificial Intelligence (UAI-01)*, pages 235–244, 2001.

[49] B.A. Huberman and T. Hogg. Phase Transitions in Artificial Intelligence Systems. *Artificial Intelligence*, 33:155–171, 1987.

[50] B. Huberman, R. Lukose, and T. Hogg. An economics approach to hard computational problems. *Science*, (265):51–54, 1993.

[51] T. Hulubei and B. O'Sullivan. Optimal Refutations for Constraint Satisfaction Problems. In *Proc. of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05)*, 2005.

[52] T. Hulubei and B. O'Sullivan. The impact of search heuristics on heavy-tailed behaviour. *Constraints*, 11(2), 2006.

[53] Y. Interian. Backdoor sets for random 3-SAT. In *Proceedings of 6th International Conference on Theory and Applications of Satisfiability Testing*, 2003.

[54] J. Hoffmann and C. Gomes and B. Selman. Structure and problem hardness: Asymmetry and DPLL proofs in SAT-based planning. In *Proceedings of the Second International Workshop on Constraint Propagation and Implementation, CP 2005*, 2005.

[55] H. Jia and C. Moore. How much backtracking does it take to color random graphs? Rigorous results on heavy tails. In M. Wallace, editor, *Proceedings of 10th International Conference on Principles and Practice of Constraint Programming (CP2004)*. Springer, 2004.

[56] H. Jia, C. Moore, and D. Strain. Generating hard satisfiable formulas by hiding solutions deceptively. In *Proceedings of AAAI 2005*. AAAI, 2005.

[57] A. Kamath, R. Motwani, K. Palem, and P. Spirakis. Tail bounds for occupancy and the satisfiability threshold conjecture. *Randomized Structure and Algorithms*, 7:59–80, 1995.

[58] H. Kautz, E. Horvitz, Y. Ruan, C. Gomes, and B. Selman. Dynamic restart policies. In *Proceedings of the 18th National Conference on AI*, pages 674–681. American Association for Artificial Intelligence, 2002.

[59] P. Kilby, J. Slaney, S. Thiebaux, and T. Walsh. Backbones and backdoors in satisfiability. In *Proceedings of the 20th National Conference on AI*. AAAI, 2005.

[60] S. Kirkpatrick and B. Selman. Critical behaviour in the satisfiability of random Boolean expressions. *Science*, 264:1297–1301, 1994.

[61] S. R. Kumar, A. Russell, and R. Sundaram. Approximating Latin square extensions. *Algorithmica*, 24:128–138, 1999.

[62] C.M. Li and Anbulagan. Heuristics based on unit propagation for satisfiability problems. In *Proceedings of the 15th IJCAI*, pages 366–371. International Joint Conference on Artificial Intelligence, 1997.

[63] M. Luby, A. Sinclair, and D. Zuckerman. Optimal speedup of Las Vegas algorithms. *Information Processing Letters*, 47:173–180, 1993.

[64] I. Lynce, L. Baptista, and J. Marques-Silva. Stochastic systematic search algorithms for satisfiability. In *Proceedings of LICS workshop on Theory and Applications of Satisfiability Testing (SAT 2001)*, 2001.

[65] I. Lynce and J. Marques-Silva. Complete unrestricted backtracking algorithms for satisfiability. In *Fifth International Symposium on the Theory and Applications of Satisfiability Testing (SAT'02)*, 2002.

[66] D. Mitchell. The resolution complexity of constraint satisfaction. In *Proceedings of 8th International Conference on Principles and Practice of Constraint Programming (CP2002)*. Springer, 2002.

[67] D. Mitchell, B. Selman, and H. Levesque. Hard and Easy Distributions of SAT Problems. In *Proceedings of the 10th National Conference on AI*, pages 459–465. American Association for Artificial Intelligence, 1992.

[68] M. Molloy and M. Salavatipour. The resolution complexity of random constraint satisfaction problems. In *Proceedings of 44th Symposium on Foundations of Computer Science (FOCS 2003)*. IEEE Computer Society, 2003.

[69] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky. 2+p SAT: Relation of typical-case complexity to the nature of the phase transition. *Random Structures and Algorithms*, 15(3-4):414–435, 1999.

[70] W. Moskewicz, C.F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. In *Proceedings of Design Automation Conference*, pages 530–535, 2001.

[71] J.A. Navarro and A. Voronkov. Generation of hard non-clausal random satisfiability problems. In *Proceedings of the 20th National Conference on AI*, pages 436–442. American Association for Artificial Intelligence, 2005.

[72] N. Nishimura, P. Ragde, and S. Szeider. Detecting backdoor sets with respect to horn and binary clauses. In *Proceedings of SAT 2004*. AAAI, 2004.

[73] P. Prosser. Binary constraint satisfaction problems: Some are harder than others. In *Proceedings of the 11th ECAI*, pages 95–99. European Conference on Artificial Intelligence, 1994.

[74] B. Selman, H. Kautz, and B. Cohen. Noise strategies for improving local search. In *Proceedings of 12th National Conference on Artificial Intelligence*, pages 337–343, 1994.

[75] P. Shaw, K. Stergiou, and T. Walsh. Arc Consistency and Quasigroup Completion. In

Proceedings of the ECAI-98 workshop on non-binary constraints, 1998.

[76] J. Slaney, M. Fujita, and M. Stickel. Automated reasoning and exhaustive search: quasigroup existence problems. *Computers and Mathematics with Applications*, 29: 115–132, 1995.

[77] J. Slaney and S. Thiébaux. On the hardness of decision and optimisation problems. In *Proceedings of the 13th ECAI*, pages 244–248. ECAI, 1998.

[78] J. Slaney and T. Walsh. Backbones in optimization and approximation. In *Proceedings of 17th IJCAI*. IJCAI, 2001.

[79] J. Slaney and T. Walsh. Phase transition behavior: from decision to optimization. In *Proceedings of the 5th International Symposium on the Theory and Applications of Satisfiability Testing, SAT 2002*, 2002.

[80] B.M. Smith. The phase transition in constraint satisfaction problems: A closer look at the mushy region. In *Proceedings of the 11th ECAI*. European Conference on Artificial Intelligence, 1994.

[81] B.M. Smith. Constructing an asymptotic phase transition in random binary constraint satisfaction. *Theoretical Computer Science*, 265:265–283, 2000.

[82] S. Szeider. Backdoor sets for DLL solvers. *Journal of Automated Reasoning*, 2006. Special issue, SAT 2005. To appear.

[83] B. Vandegriend and J. Culberson. The Gn,m phase transition is not hard for the Hamiltonian Cycle problem. *Journal of Artificial Intelligence Research*, 9:219–245, 1998.

[84] T. Walsh. The constrainedness knife-edge. In *Proceedings of the 15th National Conference on AI*. American Association for Artificial Intelligence, 1998.

[85] T. Walsh. Search in a small world. In *Proceedings of 16th IJCAI*. International Joint Conference on Artificial Intelligence, 1999.

[86] C. Williams and T. Hogg. Exploiting the deep structure of constraint problems. *Artificial Intelligence*, 70:73–117, 1994.

[87] R. Williams, C. Gomes, and B. Selman. Backdoors to typical case complexity. In *Proceedings of 18th IJCAI*. International Joint Conference on Artificial Intelligence, 2003.

[88] R. Williams, C. Gomes, and B. Selman. On the connections between backdoors, restarts, and heavy-tailedness in combinatorial search. In *Proceedings of Sixth International Conference on Theory and Applications of Satisfiability Testing (SAT-03)*, 2003.

[89] K. Xu, F. Boussemart, F. Hemery, and C. Lecoutre. A simple model to generate hard satisfiable instances. In *Proceedings of the 19th International Conference on AI*. International Joint Conference on Artificial Intelligence, 2005.

[90] K. Xu, F. Boussemart, F. Hemery, and C. Lecoutre. A simple model to generate hard satisfiable instances. In *Proceedings of IJCAI 2005*. IJCAI, 2005.

[91] K. Xu and W. Li. Exact phase transitions in random constraint satisfaction problems. *Journal of Artificial Intelligence Research*, 12:93–103, 2000.

[92] F. Zelezny, A. Srinivasan, and D. Page. Lattice-search runtime distributions may be heavy-tailed. In *Proceedings of the Twelfth International Conference on Inductive Logic Program*, 2002.

[93] H. Zhang. A random jump strategy for combinatorial search. In *Proceedings of International Symposium on AI and Math*, Fort Lauderdale, Florida, 2002.

[94] W. Zhang. Phase transitions and backbones of 3-SAT and Maximum 3-SAT. In

T. Walsh, editor, *Proceedings of 7th International Conference on Principles and Practice of Constraint Programming (CP2001)*. Springer, 2001.

[95] W. Zhang. Phase transitions and backbones of the asymmetric traveling salesman problem. *JAIR*, 21:471–497, 2004.

[96] W. Zhang and R.E. Korf. A study of complexity transitions on the asymmetric traveling salesman problem. *Artificial Intelligence*, 81(1-2):223–239, 1996.