ELSEVIER

Artificial Intelligence 81 (1996) 81-109

Artificial Intelligence

An empirical study of phase transitions in binary constraint satisfaction problems

Patrick Prosser *

Department of Computer Science, University of Strathclyde, Glasgow G1 1XH, Scotland, UK

Received April 1994; revised April 1995

Abstract

An empirical study of randomly generated binary constraint satisfaction problems reveals that for problems with a given number of variables, domain size, and connectivity there is a critical level of constraint tightness at which a phase transition occurs. At the phase transition, problems change from being soluble to insoluble, and the difficulty of problems increases dramatically. A theory developed by Williams and Hogg [44], and independently developed by Smith [37], predicts where the hardest problems should occur. It is shown that the theory is in close agreement with the empirical results, except when constraint graphs are sparse.

Keywords: Search phase transitions; Random binary constraint satisfaction problems; Empirical study of phase transition behavior in CSPs; Mushy region; Hard problems

1. Introduction

In the binary constraint satisfaction problem (CSP) we are given a set of variables, where each variable has a domain of values, and a set of constraints acting between pairs of variables. The problem is to find an assignment of values to variables, from their respective domains, such that the constraints are satisfied [5,24,26,27,40].

It has been shown that graph colouring problems exhibit a phase transition, where problems change from being easy to colour, to being hard to colour, and on to problems that obviously cannot be coloured [2]. This easy-hard-easy phase transition occurs at a critical value of connectivity. This same phenomenon has been observed for Hamiltonian paths [2], satisfiability problems (SAT) [3,13,23,28], the travelling salesman problem [14,15], and has been anticipated for problems of search more generally [43,44]. The

* Fax: +44 141 552 5330. E-mail: pat@cs.strath.ac.uk.

SSDI 0004-3702(95)00048-8

graph colouring problem can be considered as a special case of the binary CSP. Variables correspond to nodes, a constraint between a pair of variables restricts those variables to different colours, there is a uniform domain size (i.e. k, the number of colours), and one colour can conflict with only one colour in the domain of an adjacent variable (the probability of a conflict across a constraint is $1/k$). For a given number of colours there is a critical value of a single control parameter, namely connectivity, that dictates where the phase transition occurs [42].

It has been known for some time that CSPs exhibit similar behaviour [11,35]. Most notably, in Gaschnig's thesis [11], when studying randomly generated n-queen problems it was observed that

> ... the number of steps executed depends strongly on degree of constraints ... a plot of the data suggests the existence of a single sharp peak ....[1]

More recent studies [6,8,9,36] have exploited this peak in problem complexity when investigating the performance of algorithms. Consequently, this paper is not answering the question "Is there a phase transition in binary CSPs?", but rather it is providing extensive experimental evidence of its behaviour and compares that to a theory developed by Williams and Hogg [44] and independently by Smith [37]

The body of the paper is organised as follows. Section 2 describes how the random problems were created, and Section 3 gives a brief description of the algorithms used in this study. The experiments are then reported in Section 4, and in Section 5 a comparison is made with the theory. Section 6 concludes this study, and Appendix A gives a breakdown of the computational effort associated with this study.

2. Problem generation

The randomly generated (binary) CSPs are characterised by the 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 portion of the $n \cdot (n-1)/2$ possible constraints in the graph, and p_2 is the portion of the m^2 value pairs in each constraint that are disallowed by the constraint. That is, p_1 may be thought of as the density of the constraint graph, and p_2 as the tightness of constraints.

In generating a CSP $\langle n, m, p_1, p_2 \rangle$ exactly $p_1 \cdot n \cdot (n-1)/2$ constraints are randomly selected (rounded to the nearest integer), and for each constraint selected exactly $p_2 \cdot m^2$ conflicting pairs of values are selected (again, rounded to the nearest integer). This is referred to as Model B by Palmer [29], Model 1 in Bollobás [1], and is the same technique as employed by Smith [37,39].[2] It should be noted that at low values of p_1 the constraint graph may be disconnected, and when $p_2 = 0$ constraints will contain no conflicts. The CSP $\langle n, m, 1, p_2 \rangle$ corresponds to the clique K_n; $\langle n, m, p_1, 1 \rangle$ corresponds to the problem where all constraints are unsatisfiable; $\langle n, m, 0, p_2 \rangle$ is the null graph N_n; and $\langle n, m, p_1, 0 \rangle$ has no conflicts.

[1] The quote is from page 180. As further work Gaschnig proposed "E4-2: Study problem structure using parameterized random problems" and "E4-7: Relation between efficiency and number of solutions".

[2] ... and is available electronically [34].

Table 1
Translation of nomenclatures

	n	m	p_1	p_2
Dechter & Meiri	n	k	$1-p_1$	p_2
Frost & Dechter	n	k	$2c/(n(n-1))$	t
Freuder & Wallace	n	d	$2c/(n(n-1))$	$1-p$
Haselböck	n	a	d	t
Williams & Hogg	μ	b	$2a/(\mu(\mu-1))$	n/b^2

Random constraint graphs have been used in empirical studies before. For example, Dechter and Meiri [6] generated random problems $\langle n, k, p_1, p_2\rangle$, where k is domain size, p_1 is the probability that a given pair of variables will not be constrained, and p_2 is the probability that when a constraint exists a pair of values will not be allowed. In Freuder and Wallace's study of partial constraint satisfaction problems [7] CSPs are of the form $\langle n, d, c, p\rangle$, where d is domain size, c is the number of constraints, and p is the number of pairs allowed across a constraint. In Haselböck's study of interchangeabilities [20], the CSP $\langle n, a, d, t\rangle$ has domain sizes in the range 1 to a, d is the density of the constraint graph, and t is the tightness of those constraints. In Frost and Dechter's studies [8,9] CSPs are generated as $\langle n, k, c, t\rangle$ where n and k are as before, c is the number of constraints in the graph, and t is the tightness of the constraints. More recently, Williams and Hogg [44] categorise a binary CSP as $\langle \mu, b, a, n\rangle$, where μ is the number of variables, b is the uniform domain size, there are a randomly selected constraints, and for each constraint n of its b^2 possible assignments are selected as being locally nogood.

An entry in Table 1 allows a translation from one nomenclature to another. For example, Frost and Dechter's problem $\langle n, k, c, t\rangle$ translates to $\langle n, k, (2c/(n(n-1)), t\rangle$, and Williams and Hogg's problem $\langle \mu, b, a, n\rangle$ translates to $\langle \mu, b, 2a/(\mu(\mu-1)), n/b^2\rangle$.

3. The algorithms

Only systematic (i.e. complete) algorithms are suitable for this study, the reason being that the stochastic techniques would not terminate when presented with an over-constrained problem. Therefore, the algorithms used are complete and find a first solution to $\langle n, m, p_1, p_2\rangle$ or report that none exists. The algorithms are based on forward checking (FC) [19] and the conflict-directed backjumper (CBJ) [31],[3] and are described in detail in [30].

The default algorithm used in this study is forward checking with conflict-directed backjumping [30], using the directed k-consistency (DKC) extension [31] and the fail-first (FF) heuristic [19,35], and will be referred to as either FC-CBJ-DKC-FF or the default algorithm. The study also looks briefly at conflict-directed backjumping [31] with the max-cardinality (MC) heuristic [6], and this will be referred to as CBJ-MC.

[3] The most distinguishing feature of CBJ is that it jumps back to the cause of a conflict, rather than backtracking chronologically. CBJ can be thought of as a marriage of Gaschnig's backjumping routine (BJ) [11] and Dechter's graph-based backjumper (GBJ) [4].

The selection of the default algorithm was based on previous computational experience [30–33].

The standard of comparison of constraint satisfaction algorithms has traditionally been the number of consistency checks performed, i.e. the number of times pairs of variable instantiations were tested for compatibility. The objective is then to reduce the number of consistency checks. Although it can be argued that this measure can be reduced by other means, such as constraint recording [9,10,16], this may lead to a trading of maintenance of that information against the cost of re-exploration. This has lead to an argument that run time should be taken as the standard of comparison [17]. A previous study [30] showed that FC-CBJ performed less consistency checks and took less run time than other algorithms. The DKC extension and FF heuristic has lead to a further reduction in run time. Since the main thrust of this study is to examine problems, not algorithms, the fastest and most reliable algorithm was chosen, thus allowing the largest amount of experimentation possible.

The algorithms were initially encoded in Sun Common Lisp, SCLisp version 4.0, and run on SUN SPARCstation IPCs with 16MB of memory. In order to increase the rate that experiments could be performed the algorithms were later recoded in C [25]. The Common Lisp implementation of the algorithms and problem generator are available electronically (see [34]).

4. The experiments

The first part of this study looks at what happens when we increase the tightness of constraints (p_2), increase the density of the constraint graph (p_1), and change algorithms (from FC-CBJ-DKC-FF to CBJ-MC). The effect of the number of variables (n) and the domain size (m) is then investigated. The last subsection relates some of the computational experiences of the experiments.

4.1. Constraint tightness, graph density, and choice of algorithm

Problems were generated with 20 variables and a uniform domain size of 10. Collectively, these problems will be referred to by the tuple $\langle 20, 10\rangle$. The density of the constraint graph, p_1, was varied from 0.1 to 1.0 in steps of 0.1, and for each value of p_1 the constraint tightness p_2 was varied from 0.01 to 0.99 in steps of 0.01.[4] At each setting of $\langle 20, 10, p_1, p_2\rangle$ 100 problems were generated. The search algorithm was then applied to each problem. Numerous statistics were gathered, such as the search effort expended in terms of consistency checks, the number of backtracks, CPU time used, and whether the problem was soluble or insoluble. In total 99,000 $\langle 20, 10\rangle$ problems were investigated.

Fig. 1 shows what happens when p_2 is varied. Fig. 1(a) shows search effort, measured as average number of consistency checks for 100 problems at $\langle 20, 10, 0.5, p_2\rangle$, against constraint tightness (p_2). As p_2 increases from 0.01 to 0.27 the search effort falls. This

[4] Clearly there is no point investigating problems with $p_2 = 0$ or problems with $p_2 = 1.0$.

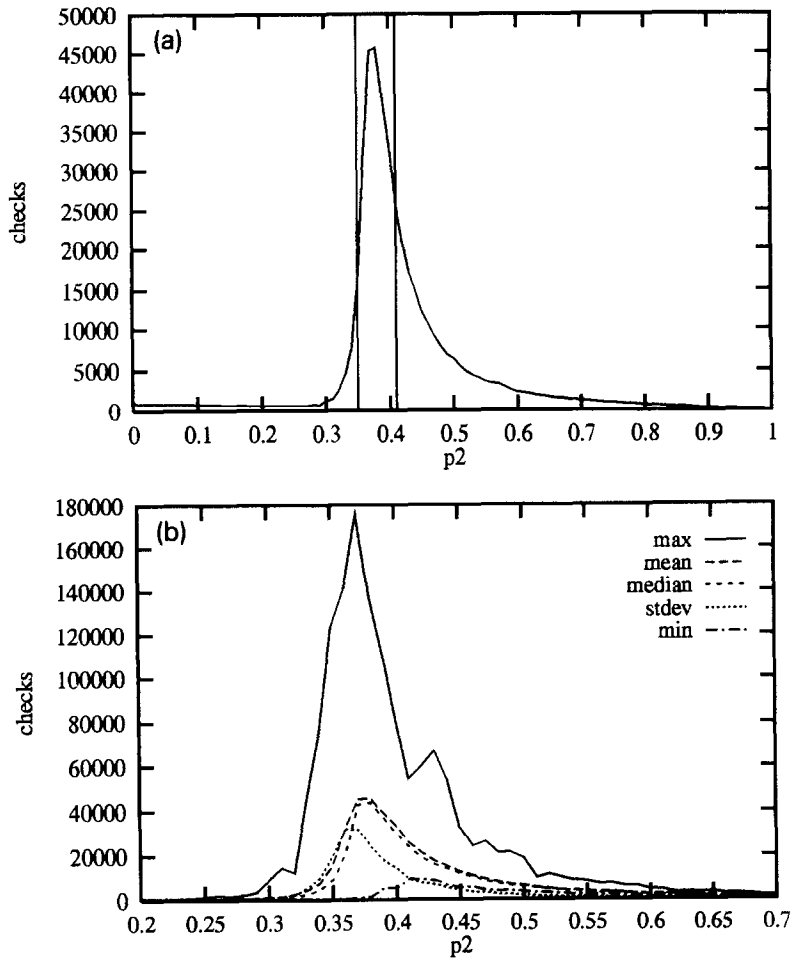

Fig. 1. Search effort against p_2 for $\langle 20, 10, 0.5 \rangle$.

is due to the increasing effectiveness of domain filtering as p_2 increases. The onset of backtracking is at $p_2 \approx 0.25$, and when $p_2 \approx 0.3$ the search effort begins to increase rapidly, reaching a maximum at $p_2 \approx 0.38$, and then falls away.[5] The vertical bar on the left-hand side is the largest value of p_2 at which all problems were soluble ($p_2 = 0.35$) and the right-hand vertical bar is the smallest value of p_2 at which all problems were insoluble ($p_2 = 0.41$). Therefore all problems to the left of the first vertical are soluble, and all problems to the right of the second vertical are insoluble. In the region between (i.e. $0.35 < p_2 < 0.41$) there is a mixture of soluble and insoluble problems, and Smith has referred to this as the *mushy region* [37].[6] It is in this region that the average search effort is maximal.

[5] At $p_2 = 0.37$, the average effort is 45,262 checks and, at $p_2 = 0.38$, the average effort is 45,710 checks.

[6] An analogy is drawn from the process of freezing, where an intermediate state is reached, neither liquid nor solid.

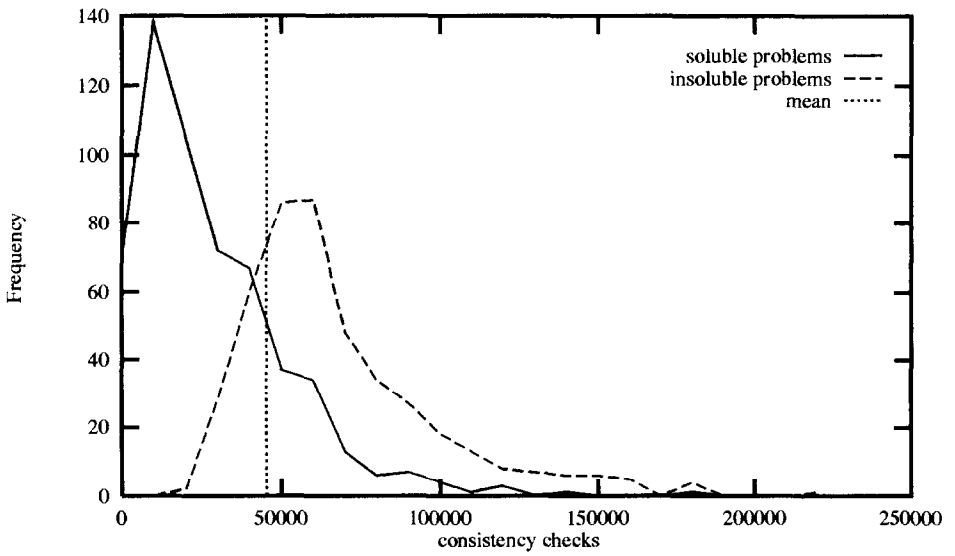

Fig. 2. Frequency distribution of problem difficulty for $\langle 20, 10, 0.5, 0.37 \rangle$.

Fig. 1(b) is again the $\langle 20, 10, 0.5 \rangle$ problems, but five curves are drawn, namely the median, the mean, the minimum, maximum, and the standard deviation. There are a number of points worth noting. Firstly, the location of the peak in the median search effort (43,945 checks) coincides with the peak in the mean (45,710 checks), at $p_2 \approx 0.38$. This closely corresponds to the crossover point, where 50% of the problems were soluble. At $p_2 = 0.37$, 56% of the problems were soluble, and at $p_2 = 0.38$, 26% were soluble. Secondly, the peak in the maximum search effort (175,409 checks) occurs before the peak in the mean search effort.

That is, the single most difficult problem was encountered at $p_2 = 0.37$, inside the mushy region, but to the left of the peak in the average and median search effort. Furthermore, the hardest problem was insoluble. This is a pattern that repeats itself, with the hardest problem tending to be insoluble and close to the lower boundary of the mushy region. The minimum search effort peaks close to the upper boundary of the mushy region, at $p_2 = 0.40$. This suggests that given a sample of problems in the mushy region, the insoluble problems might be harder than soluble problems. Fig. 2 and Table 2 tend to support this conjecture. Finally, we see that the variation in the search effort (i.e. contour *stdev*) is at a maximum where the very hardest problem occurred, at $p_2 = 0.37$.

Fig. 2 shows a frequency distribution of problem difficulty (x-axis) against number of problems at that difficulty (y-axis) for 1,000 problems at $\langle 20, 10, 0.5, 0.37 \rangle$, i.e. we see a slice through the curve in Fig. 1(a) close to where the mean and the median search effort reaches a maximum. Two frequency distributions are given, one for the 440 insoluble problems, the other for the 560 soluble problems in the sample. The vertical bar shows the average problem difficulty for the whole sample. The insoluble problems were on average harder than the soluble problems, the easiest insoluble problem was

Table 2
A closer look at the mushy region of ⟨20, 10, 0.5⟩

n	m	p_1	p_2		#problems	μ	σ	median	min	max
20	10	0.5	0.35	all	1,000	17,291	27,104	9113	487	438,498
20	10	0.5	0.35	soluble	990	16,149	24,027	9,024	487	438,498
20	10	0.5	0.35	insoluble	10	130,338	58,317	123,648	32,519	244,938
20	10	0.5	0.36	all	1,000	32,319	33,780	21,049	480	268,290
20	10	0.5	0.36	soluble	883	25,212	24,800	17,049	480	190,682
20	10	0.5	0.36	insoluble	117	86,127	38,679	74,370	33,113	268,290
20	10	0.5	0.37	all	1,000	45,353	33,187	42,084	511	224,022
20	10	0.5	0.37	soluble	560	27,523	23,762	21,147	511	184,089
20	10	0.5	0.37	insoluble	440	68,046	30,906	58,972	22,492	224,022
20	10	0.5	0.38	all	1,000	46,997	24,223	44,034	561	211,627
20	10	0.5	0.38	soluble	228	27,412	20,393	22,952	561	103,969
20	10	0.5	0.38	insoluble	722	52,781	22,147	48,538	16,249	211,627

much harder than the easiest soluble problem, and the hardest problem was insoluble. Both distributions are skewed with a few extremely hard problems out to the right, and the distribution is more dispersed for soluble than insoluble problems. Fig. 2 gives some explanation as to why the curve of the minimum number of consistency checks in Fig. 1(b) peaks at the upper boundary of the mushy region. As p_2 increases the number of soluble (relatively easy) problems decreases and the number of insoluble (relatively hard) problems increases. At the upper boundary of the mushy region all problems are insoluble (by definition), consequently the easiest problems in the sample continue to be relatively hard.

These experiments were repeated with $p_2 = 0.35$ (just entering the mushy region), $p_2 = 0.36$ (just before the average search effort peaks), and $p_2 = 0.38$ (just before leaving the mushy region), and the results are shown in Table 2. There are three rows for each problem, showing statistics on the entire sample of 1,000 problems, and statistics on the soluble and insoluble problems in that sample. The first five columns identify the problems and if those problems are the entire sample, the soluble problems, or the insoluble problems. The next column (#problems) gives the number of problems in that sample, then the average number of consistency checks for the sample (μ), the standard deviation (σ), the median, the easiest problem (min), and the hardest problem (max). In the mushy region, as p_2 increases the number of soluble problems falls and the average difficulty of those problems increases, and then gradually falls away. For the insoluble problems, as p_2 increases the number of insoluble problems increases and difficulty of those problems decreases. Insoluble problems are on average harder than their soluble counterpart, therefore as p_2 increases the average difficulty is dominated by an increasing number of insoluble problems. This ultimately leads to all problems being insoluble, and a gradual decay in the average difficulty.

Looking again at Table 2 it is interesting to note that as a result of increasing the sample size, from 100 to 1,000, we have found harder problems. Comparing the contour of maximum search effort in Fig. 1(b) with the max column in Table 2 we see that at $p_2 = 0.37$ the value has increased from 175,409 checks to 224,022 checks. Furthermore,

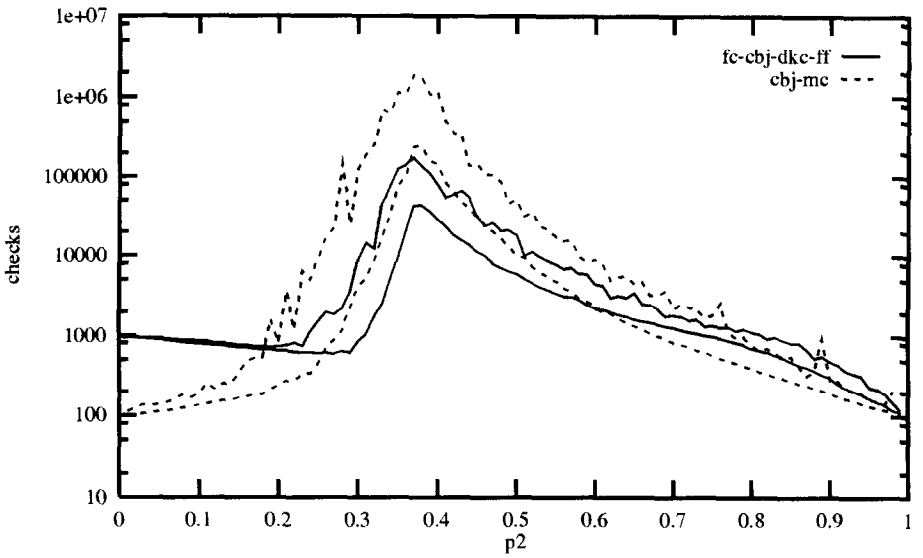

Fig. 3. $\langle 20, 10, 0.5 \rangle$ for two different algorithms, FC-CBJ-DKC-FF and CBJ-MC, showing for each algorithm the maximum and median search effort.

we now see that the very hardest problem is soluble and occurred at $p_2 = 0.35$ taking 438,498 checks. This tends to add weight to the observations made by Hogg and Williams [21].

The location of the peak search effort appears to be algorithm-independent. Fig. 3 shows the performance of conflict-directed backjumping with the max-cardinality heuristic (CBJ-MC) and the default algorithm (FC-CBJ-DKC-FF). Two curves are drawn for each algorithm, namely the median search effort and the maximum search effort. Note also, that the y-axis is on a logarithmic scale. Both algorithms encounter the very hardest problems at the same value of p_2 (i.e. $p_2 = 0.37$), the medians peak at the same position (i.e. $p_2 = 0.38$), and both algorithms have similar signatures (the curves are of similar shape). The default algorithm is on average an order of magnitude better than CBJ-MC on the hard $\langle 20, 10, 0.5 \rangle$ problems. The 100 problems at each value of p_2 in the mushy region were analysed, and again it was seen that CBJ-MC found the insoluble problems significantly harder than the soluble problems, and as p_2 increased the difficulty of the soluble problems increased and the difficulty of the insoluble problems decreased.

Therefore, the phenomenon described in Table 2 and Fig. 2 appears to be a feature of the problem and not the algorithm (i.e. algorithm-independent). Fig. 4 shows what happens when the density of the constraint graph is increased. Fig. 4 shows the search effort for the $\langle 20, 10, 1.0 \rangle$ problems, and is comparable to Fig. 1 (i.e. $\langle 20, 10, 0.5 \rangle$). On the top (Fig. 4(a)) the average search effort is plotted along with the boundaries of the mushy region, and on the bottom (Fig. 4(b)) the maximum, mean, median, standard deviation, and minimum search effort are shown. The curves have a similar signature, but Fig. 4(a) is more pronounced than Fig. 1(a) and is translated left to

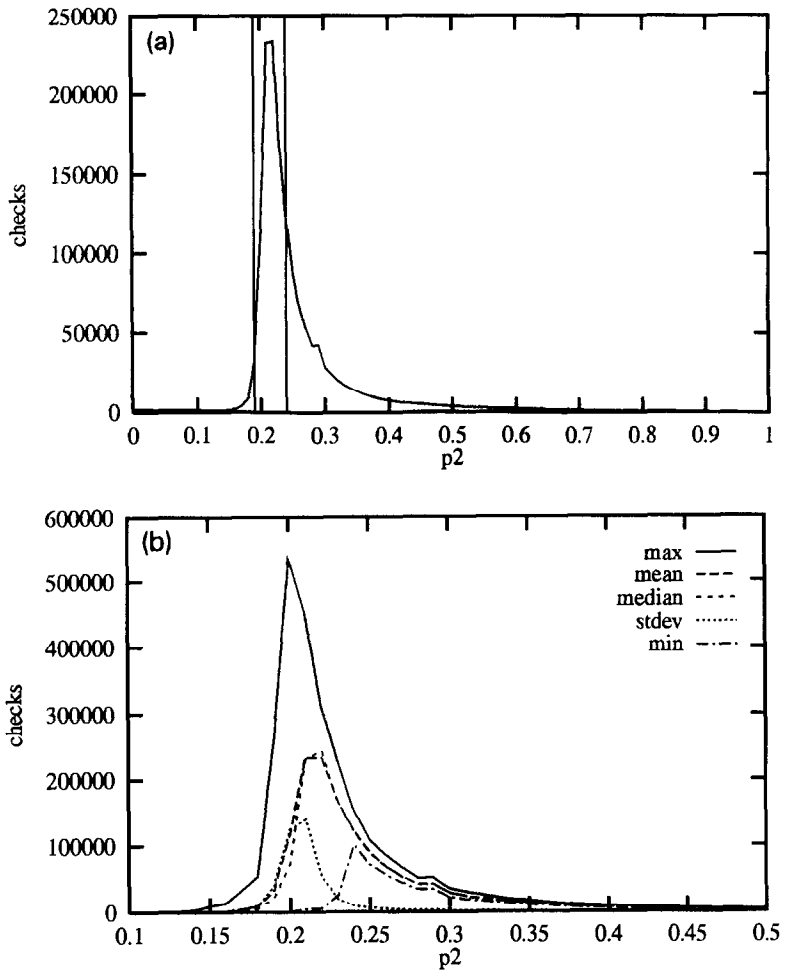

Fig. 4. Search effort against p_2 for $\langle 20, 10, 1.0 \rangle$.

smaller values of p_2. It should be noted that the scale on the y-axis of Fig. 4(a) goes to 250,000 whereas in Fig. 1(a) the scale goes to 50,000. That is, as the density of the constraint graph increases (and all else remains the same) the difficulty of the problems increases [33]. The peak of the average search effort (233,641 checks) occurred at $p_2 = 0.22$, and coincides with the peak in the median (243,170 checks). Again, the very hardest problem (535,616 checks, and insoluble) occurred before the peak in the mean, at $p_2 = 0.20$, and the standard deviation peaks at $p_2 = 0.21$.

The maximum value of p_2 for soluble $\langle 20, 10, 1.0 \rangle$ problems is 0.19, and the minimum value where all problems are insoluble is $p_2 = 0.24$; i.e. the mushy region for $\langle 20, 10, 1.0 \rangle$ is 0.04 wide, compared to 0.05 for $\langle 20, 10, 0.5 \rangle$. Again, the minimum search effort peaked at the upper boundary of the mushy region. The distribution of problem difficulty at $\langle 20, 10, 1.0, 0.21 \rangle$ was investigated, as in Fig. 2 and Table 2 above.

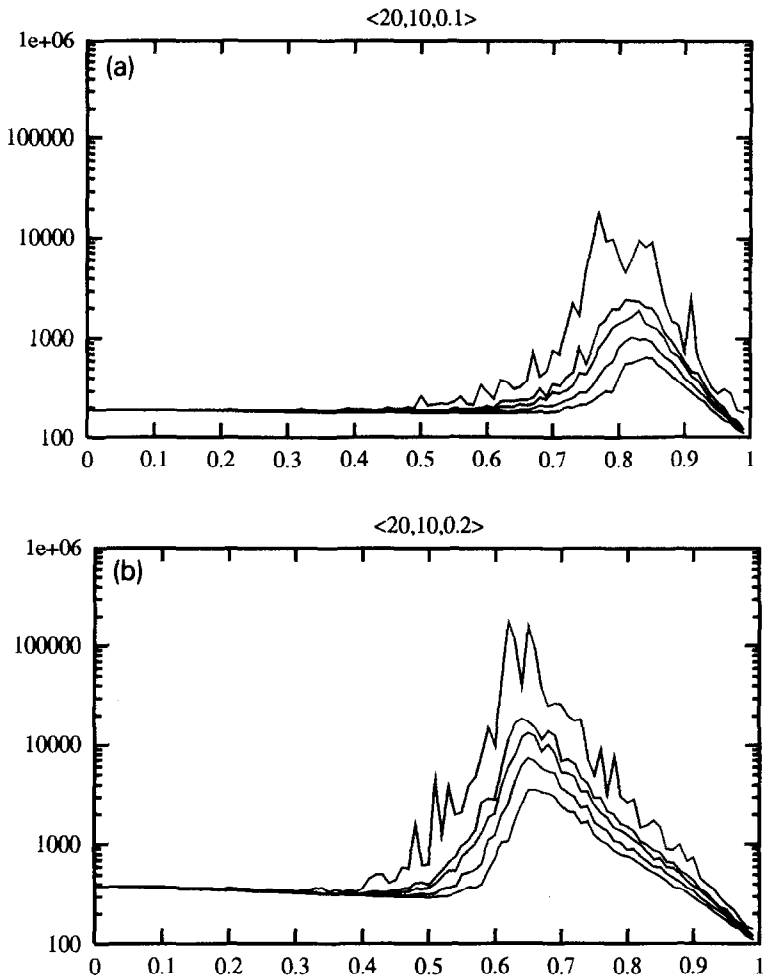

Fig. 5. Search effort for $\langle 20, 10, p_1 \rangle$, against p_2, with p_1 varying from (a) 0.1, (b) 0.2, (c) 0.3, to (d) 0.4.

$\langle 20, 10, 1.0, 0.21 \rangle$ was selected instead of $\langle 20, 10, 1.0, 0.22 \rangle$ because only 10% of the problems were soluble at $p_2 = 0.22$, whereas 62% were soluble at $p_2 = 0.21$. Again, the insoluble problems were on average harder (370,598 checks) than the soluble problems (151,734 checks), the easiest insoluble problem (291,471 checks) was harder than the easiest soluble problem (1,052 checks), and the hardest problem was insoluble (469,453 checks).

Up to this point, the analysis has tended to rely on the average search effort of an ensemble of problems. This is, arguably, a reasonable thing to do as it gives us an indication of the amount of effort that would be required to address a sample of $\langle n, m, p_1, p_2 \rangle$ problems. However, using the mean in isolation exposes only some of the structure of the phase transition phenomena. In order to get a better understanding we must look at the distribution about the mean (as in [12,21]).

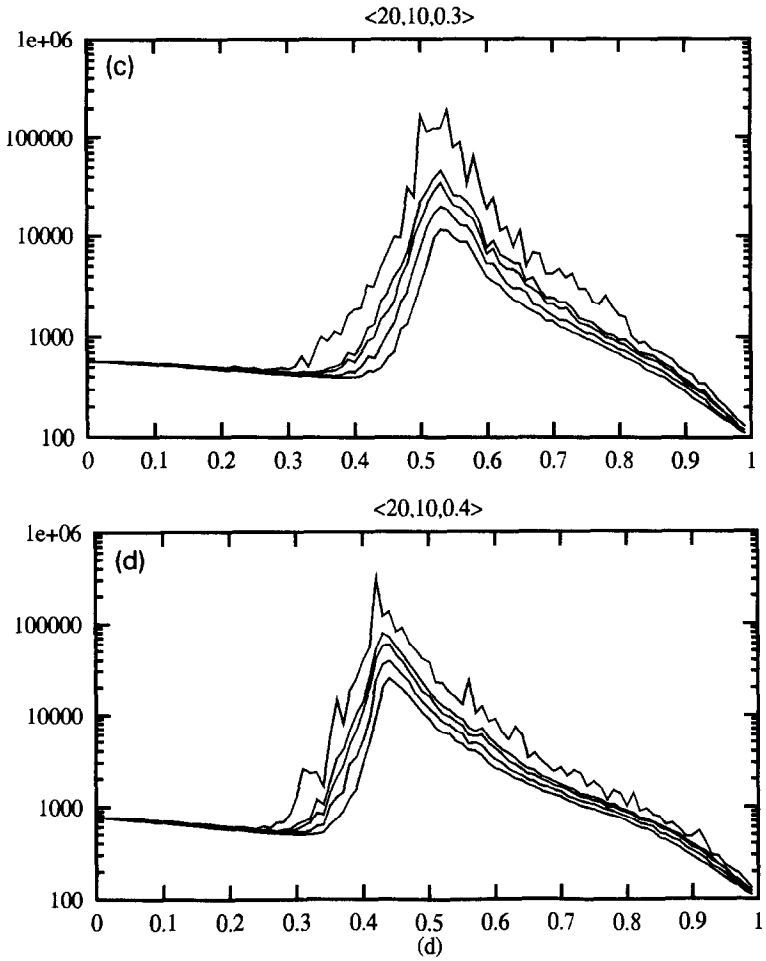

Fig. 5 — continued.

High variability in search effort was observed when the density of the constraint graphs was low. This is shown in Fig. 5. Each of the four graphs plots contours for the following percentiles: 100% (the very hardest problem), 95%, 90%, 75%, and 50% (the median). For example, the 90% contour gives the cost that was exceeded by only 10% of the problems in the sample. Figs. 5(a) is for $\langle 20, 10, 0.1 \rangle$, Fig. 5(b) is for $\langle 20, 10, 0.2 \rangle$, Fig. 5(c) is for $\langle 20, 10, 0.3 \rangle$, and Fig. 5(d) is for $\langle 20, 10, 0.4 \rangle$. The x-axis is p_2 and the y-axis is search effort, and is a logarithmic scale. At low values of p_1, most notably when $p_1 = 0.1$, there is high variability in search effort, and this occurs well in advance of the peak in the median and mean search effort. For example, in Fig. 5(a) the 100% contour peaks at $p_2 = 0.77$, whereas the 50% contour peaks at $p_2 = 0.84$. As the density of the constraint graph increases (going from Fig. 5(a) to 5(d)) this effect diminishes, and we see the locations of the peaks in the contours gradually moving together. Similarly, the mushy region narrows. In Fig. 5(a) the boundaries of the mushy

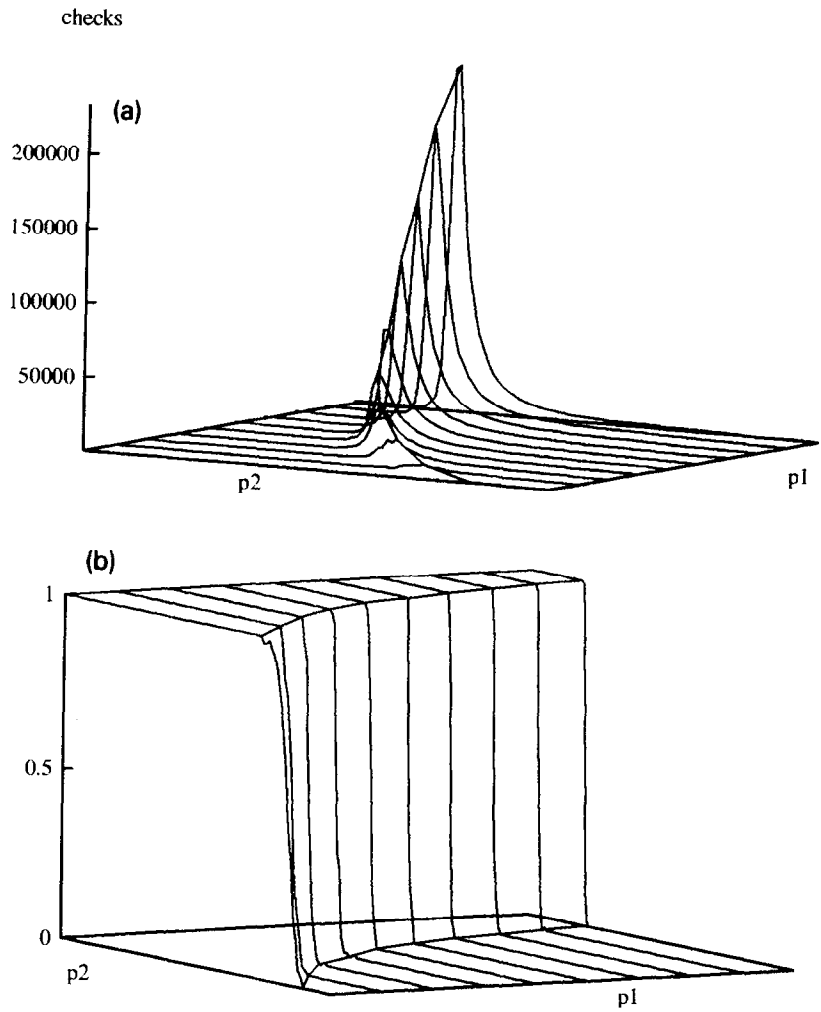

Fig. 6. A map of search effort (a) and solubility (b) for all ⟨20, 10⟩ problems.

region are respectively 0.74 and 0.91 (i.e. 0.13 wide), and in Fig. 5(d) the boundaries are 0.40 and 0.48 (i.e. 0.08) wide. This appears to bottom out at a width of 0.05 when $p_1 = 0.5$, remaining pretty much the same right up to $p_1 = 1.0$.

This high variability in search effort has been observed in SAT, graph colouring, travelling salesman problems, and binary constraint satisfaction problems by others [12, 14, 15, 21, 39]. In most of these studies these exceptionally hard problems (referred to as EHPs by Smith [39] and Hogg and Williams [21]) have often been large enough to influence the location of the peak in the mean search effort. In this study that has not been the case, and the peak in the mean and the median have generally coincided. This might be due to a number of things. Firstly, these EHPs are by definition rare, and one must have a relatively large sample in order to encounter them. Maybe a

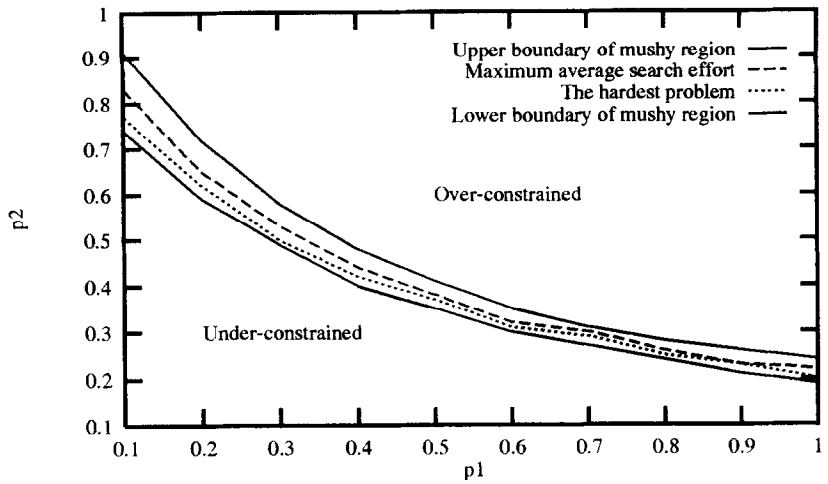

Fig. 7. The mushy regions for all ⟨20, 10⟩ problems.

sample size of 100, or even 1,000, is too small for them to have a significant effect. A second possible reason is that EHPs are, to some degree, an algorithm-dependent phenomenon, such that they occur more frequently when using chronological backtrackers. This study used algorithms that jump back, consequently fewer EHPs might be encountered.

From the previous figures it becomes apparent that for a given value of p_1 there is a value of p_2, call it $p_{2\text{crit}}$, where the average search effort is maximal. High values of p_1 (dense constraint graphs) have low values of $p_{2\text{crit}}$ (require relatively loose constraints to become hard to solve), and low values of p_1 (sparse constraint graphs) have high values of $p_{2\text{crit}}$ (tight constraints). As p_1 increases the peak average search effort increases and the definition of the curve becomes more pronounced (compare Figs. 1 and 4), and when p_2 increases beyond $p_{2\text{crit}}$ the search effort falls away slowly.

The mushy region has been defined by two boundaries. The lower boundary $mushy_{\text{lwb}}$ is taken as the largest value of p_2 where all problems were soluble, and the upper boundary $mushy_{\text{upb}}$ is taken as the smallest value of p_2 where all problems were insoluble.[7] The hardest problems occurred in the mushy region, and as p_1 increased the width of the mushy region (i.e. $mushy_{\text{upb}} - mushy_{\text{lwb}}$) decreased. When constraint graphs were sparse (and in particular when $p_1 = 0.1$) there was high variability in search effort well before the crossover point.

Fig. 6(a) shows a three-dimensional view of all ⟨20, 10⟩ problems and corresponds to the application of FC-CBJ-DKC-FF to 99,000 randomly generated problems. The x-axis is p_2 (constraint tightness), the y-axis is p_1 (density of the constraint graph), and the z-axis is search effort measured as mean number of consistency checks performed over

[7] This is a pragmatic definition of those boundaries. For example, if enough samples are taken at some value of $p_2 \leqslant mushy_{\text{lwb}}$ some insoluble problems will be discovered. Even if they were not created by the problem generator, they might be hand-crafted with little effort. The same holds true for $mushy_{\text{upb}} \leqslant p_2$; soluble problems may be discovered or created manually.

100 problems at a given point $\langle 20, 10, p_1, p_2 \rangle$. Going into the page p_1 increases from 0.1 to 1.0 in steps of 0.1, and moving left to right across the page p_2 increases from 0.01 to 0.99 in steps of 0.01. The peaks of the individual $\langle 20, 10, p_1 \rangle$ curves have been joined to enhance the picture.

There is a ridge, and it tracks from right to left and climbs going into the page. Fig. 6 might be considered as a map of the difficulty of all randomly generated $\langle 20, 10 \rangle$ problems. Fig. 6(b) shows where the soluble and insoluble problems exist for $\langle 20, 10 \rangle$. As in Fig. 6(a), the x-axis is p_2, the y-axis (going into the page) is p_1, and the z-axis is p_{sol} (where p_{sol} is the ratio of soluble problems over the number of problems examined). There is a plateau where all problems are soluble ($p_{sol} = 1.0$), and then an abrupt fall to a lower plateau where all problems are insoluble ($p_{sol} = 0.0$). Where Fig. 6(a) looks like a ridge, Fig. 6(b) looks like a cliff face.

Fig. 7 brings together the information displayed in Figs. 1–6. The x-axis is p_1 and the y-axis is p_2. The lower curve is the lower boundary of the mushy region for a given value of p_1, and the top curve is the upper boundary of the mushy region for a given value of p_1. The region between these two curves is the mushy region for all $\langle 20, 10 \rangle$ problems, the bold broken curve shows where the maximum average search effort occurred, and the faint broken line shows were the very hardest problem was encountered. Problems that exist below the lower curve tend to be easy and soluble, and problems that exist above the upper curve tend to be easy and insoluble.

4.2. *Increasing the number of variables*

The number of variables was increased from 20 to 30, the domain size was held at 10, p_1 was varied in steps of 0.1 from 0.1 to 1.0, at each setting $\langle 30, 10, p_1 \rangle$ p_2 was varied from 0.01 to 0.99 in steps of 0.01, and at each setting $\langle 30, 10, p_1, p_2 \rangle$ 100 problems were examined (i.e. 99,000 problems were examined). Fig. 8(a) shows a three-dimensional view of all $\langle 30, 10 \rangle$ problems and is comparable to Fig. 6 (i.e. $\langle 20, 10 \rangle$). Both figures have the same topography, namely a ridge, but $\langle 30, 10 \rangle$ is translated left to smaller values of p_2 and is higher and steeper (note the scale in Fig. 8).

The number of variables (n) was then increased from 20 to 60 in steps of 10, domain size was held at 10, p_1 was held at 0.1, p_2 was varied from 0.01 to 0.99 in steps of 0.01, and 100 problems were generated at each value of $\langle n, 10, 0.1, p_2 \rangle$ (i.e. 49,500 problems were analysed). These results are shown in Figs. 9 and 10.

Fig. 9 shows search effort (on a logarithmic scale) against constraint tightness for $\langle n, 10, 0.1 \rangle$, with n varying from 30 to 60 in steps of 10. In each sub-picture five contours are drawn, for the percentiles 50%, 75%, 90%, 95%, and 100%, and are similar to the sub-pictures in Fig. 5 (and the graph for $\langle 20, 10, 0.1 \rangle$ is shown in Fig. 5(a)). We see that at low values of n the 100% contour is noisy, and hard problems tend to occur at relatively low values of p_2, away from the peak of the lower percentile curves. This behaviour tends to disappear when n increases, and the locations of the peaks tend to coincide.

In Fig. 10(a) the maximum search effort (y-axis, a logarithmic scale) is plotted against the number of variables (x-axis). Six contours are plotted, for the range of percentiles and the mean. Fig. 10(a) should be interpreted as follows. Looking at the

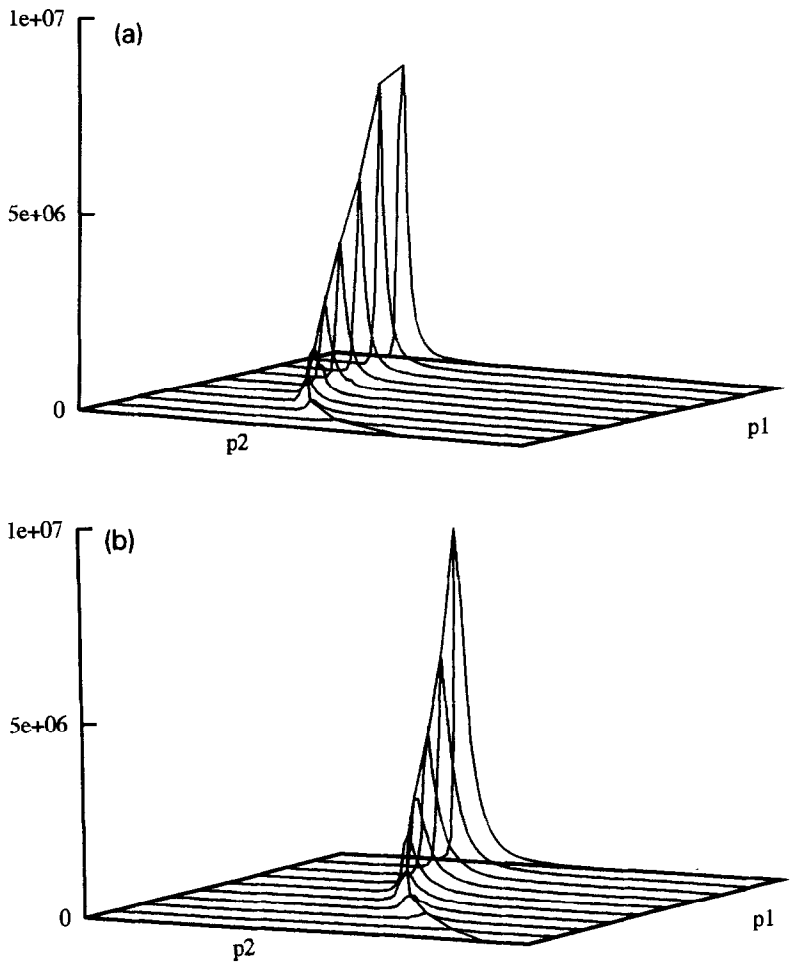

Fig. 8. A map of search effort for (a) all $\langle 30, 10\rangle$ problems, and (b) all $\langle 20, 20\rangle$ problems.

curve for the mean (the broken line), we see the maximum average search effort for a given value of n, and looking at the 100% contour we see the search effort associated with the very hardest problem at a given value of n. Fig. 10(b) shows the location of the complexity peaks plotted in Fig. 10(a) and the boundaries of the mushy region, and should be interpreted as follows. Selecting the contour for the hardest problem (for example), we pick off the value of p_2 (y-axis) for a given value of n (x-axis) where the very hardest $\langle n, 10, 0.1\rangle$ occurred, and taking the curve for the average search effort (for example) we pick off the p_2 value where the average search effort was a maximum for a given value of n. Fig. 10(a) shows that as n increases the search effort increases exponentially. This is what should be expected, since the worst case complexity of a CSP is of the order $O(m^n)$. That is, we can consider a CSP as an n-digit number to the base m, and we must find a number that satisfies some property. In the worst case

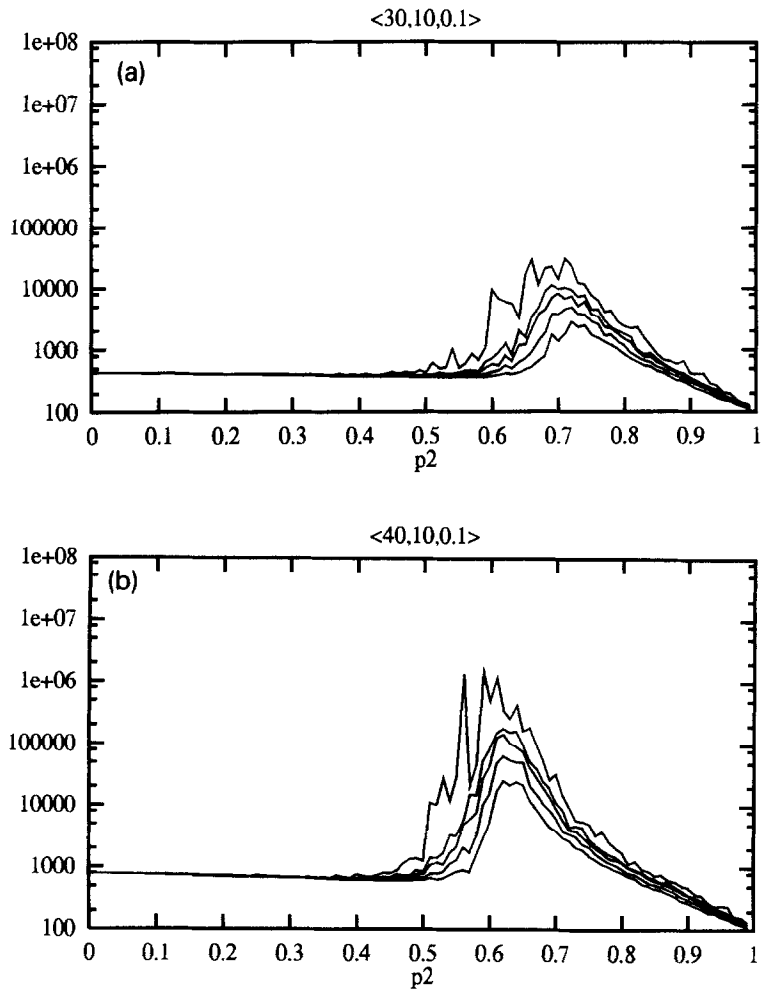

Fig. 9. The effect of increasing the number of variables, in (a) $\langle 30, 10, 0.1\rangle$, (b) $\langle 40, 10, 0.1\rangle$, (c) $\langle 50, 10, 0.1\rangle$, and in (d) $\langle 60, 10, 0.1\rangle$.

we might have to examine all possible m^n numbers. We also see that as n increases, and all else remains constant, the absolute variance in search effort increases (i.e. the contours for the various percentiles are relatively straight and parallel, and the y-axis is a logarithmic scale).

There are a number of notable features in Fig. 10(b). Firstly, as n increases, and the domain size and graph density remain constant, the complexity peaks occur at lower values of p_2 (and this is also shown in Fig. 9). Secondly, as n increases (and all else remains the same) the width of the mushy region decreases. Finally, the location of the mean and the median nearly always coincide, and the very hardest problems generally occur before the peak in the mean, and within the mushy region (again, also shown in Fig. 9). It might be worth stating that these observations must be made with some

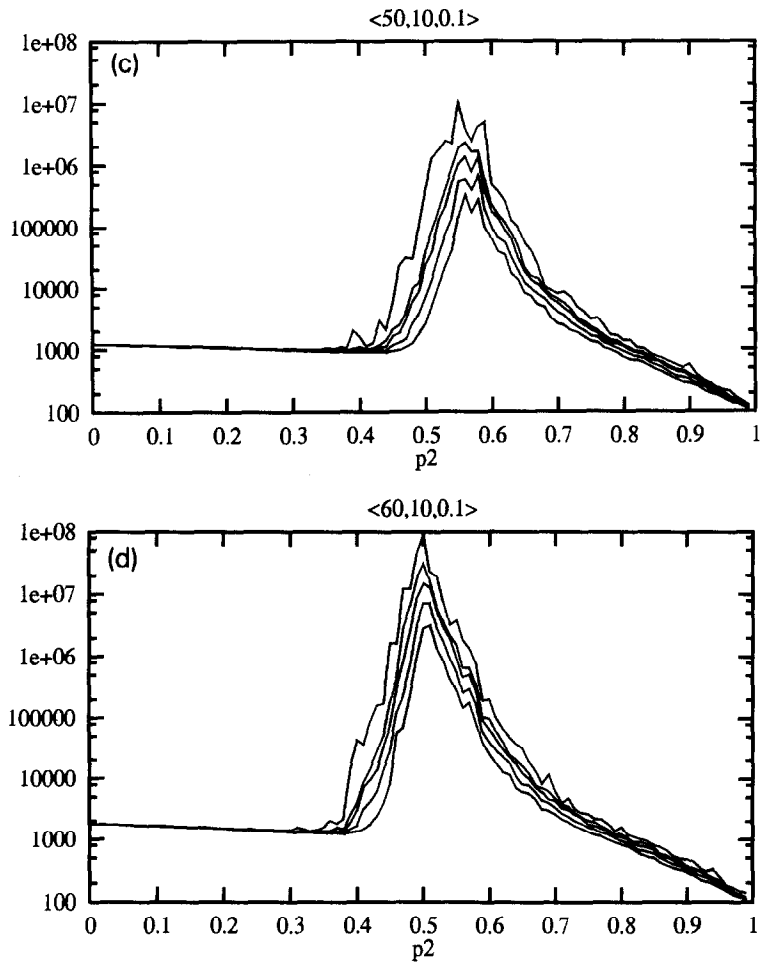

Fig. 9 — continued.

caution. Although this set of experiments involved some tens of thousands of $\langle n, 10, 0.1\rangle$ problems and hundreds of hours of CPU time, in this view of the data the contours pass through only 5 values of n.

4.3. *Increasing the domain size*

The domain size was increased from 10 to 20, and the same set of experiments was repeated (i.e. those depicted in Fig. 6 for $\langle 20, 10\rangle$) but for $\langle 20, 20\rangle$. Fig. 8(b) shows a three-dimensional view of all the $\langle 20, 20\rangle$ problems. Again we see a familiar landscape; a ridge that tracks right to left climbing as it goes. However the ridge is translated right relative to $\langle 20, 10\rangle$ and is much steeper and higher.

The domain size (m) was then increased from 10 to 50 in steps of 10, n was held at 20, p_1 held at 0.5, p_2 was varied from 0.01 to 0.99 in steps of 0.01, and 100 problems

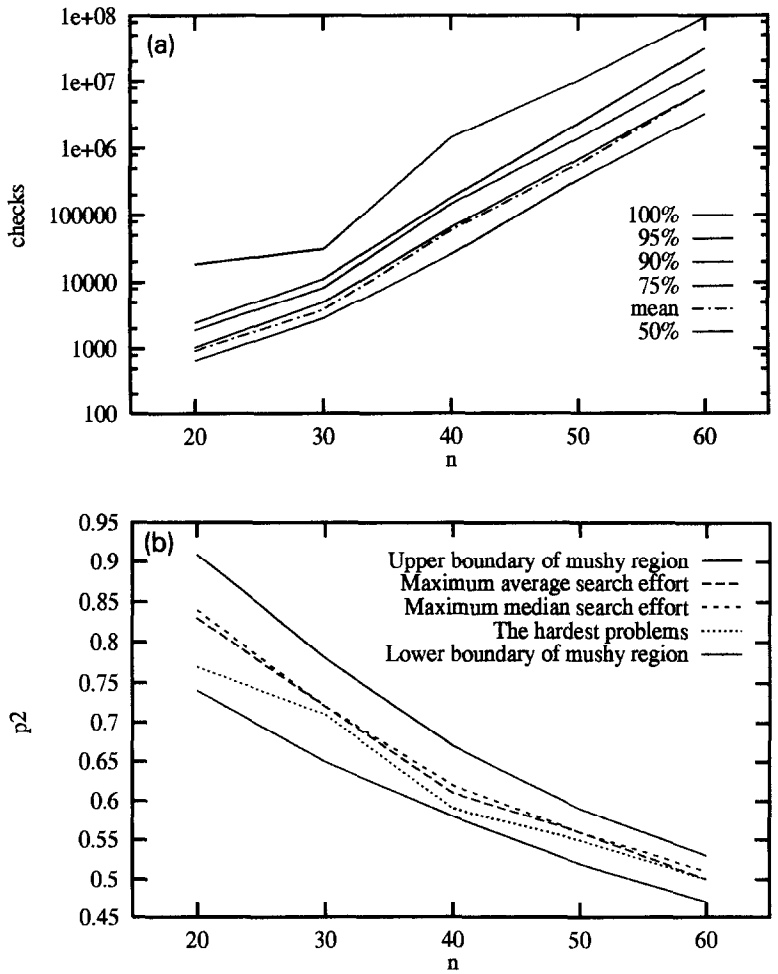

Fig. 10. Search effort against number of variables, n, for $\langle n, 10, 0.1\rangle$ and locations of complexity peaks and the boundaries of the mushy region.

were generated at each value of $\langle 20, m, 0.5, p_2\rangle$ (i.e. 49,500 problems were analysed). The results of those experiments are shown in Figs. 11 and 12.

Fig. 11 is of the same format as Figs. 5 and 9, and shows search effort against constraint tightness for $\langle 20, 20, 0.5\rangle$, $\langle 20, 30, 0.5\rangle$, $\langle 20, 40, 0.5\rangle$, and $\langle 20, 50, 0.5\rangle$. These problems tends to be quite well behaved, with the location of the peaks (of the percentiles) being relatively close together.

Fig. 12 is of the same format as Fig. 10 except the x-axis is now m, the domain size. In Fig. 12(a) we see that as m increases the search effort increases at a rate that is less than exponential, and this is what should be expected. Essentially the contours have the same shape as the function $y = m^{20}$.

In Fig. 12(b) we see clearly that as domain size increases the corresponding value

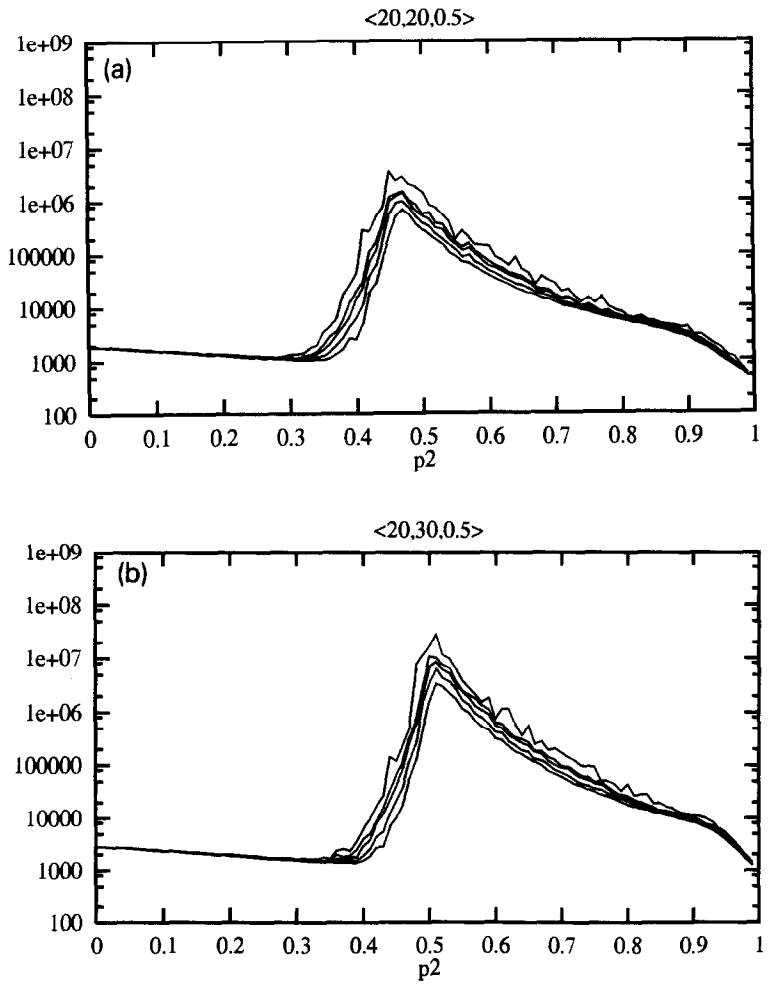

Fig. 11. The effect of increasing domain size, in (a) $\langle 20,20,0.5\rangle$, (b) $\langle 20,30,0.5\rangle$, (c) $\langle 20,40,0.5\rangle$, and in (d) $\langle 20,50,0.5\rangle$.

of p_2 for the complexity peaks also increases. This is rather intuitive, as we should expect that as the number of values available to variables increases, the tightness of the constraints has to increase in order to make the problems more difficult to satisfy. Furthermore, it appears that the width of the mushy region does not change significantly as domain size varies. There is insufficient data to determine unequivocally if the hardest problems tend to coincide with the maximum average search effort, and if the variation in search effort diminishes as domain size increases. Fig. 12(b) shows one data point where the hardest problem coincides with the peak in the average (also shown in Fig. 11(c)), and one data point where the hardest problem actually occurs after the peak in the average (and this point actually corresponds to the hardest problem encountered in all of the experiments, see Figs. 11(c) and 12(a)). These two data points could just

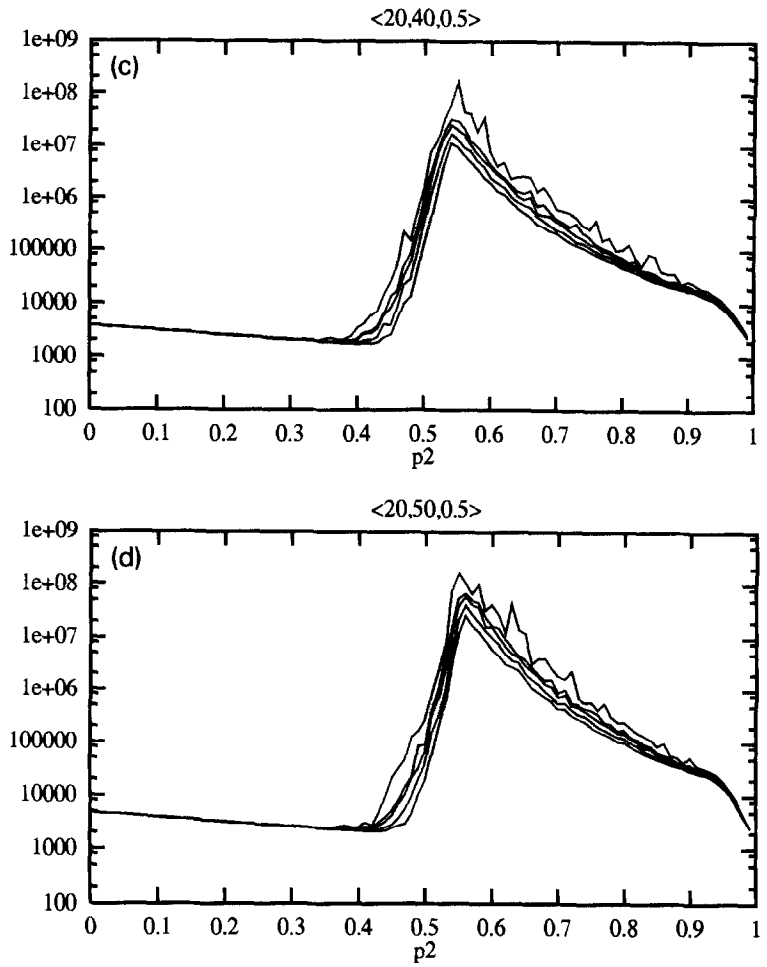

Fig. 11 — continued.

be noise, and one initially suspects so. In order to resolve this we would require either larger sample sizes at each data point or more data points. Unfortunately, that would be extremely expensive to do. [8]

4.4. Computational experience

A limited amount of experiments were performed using a range of algorithms, in particular the CBJ-based algorithms, and chronological backtrackers. There were cases

[8] The $\langle 20, 50, 0.5 \rangle$ experiments were the most expensive experiments performed, taking in excess of 359 hours, about 15 days. The 100 problems at $\langle 20, 50, 0.5, 0.56 \rangle$ alone took in excess of 79 hours. The algorithms were re-coded in C for these experiments, and up to 150 workstations were used at any one time. More information on computational effort is given in Appendix A.

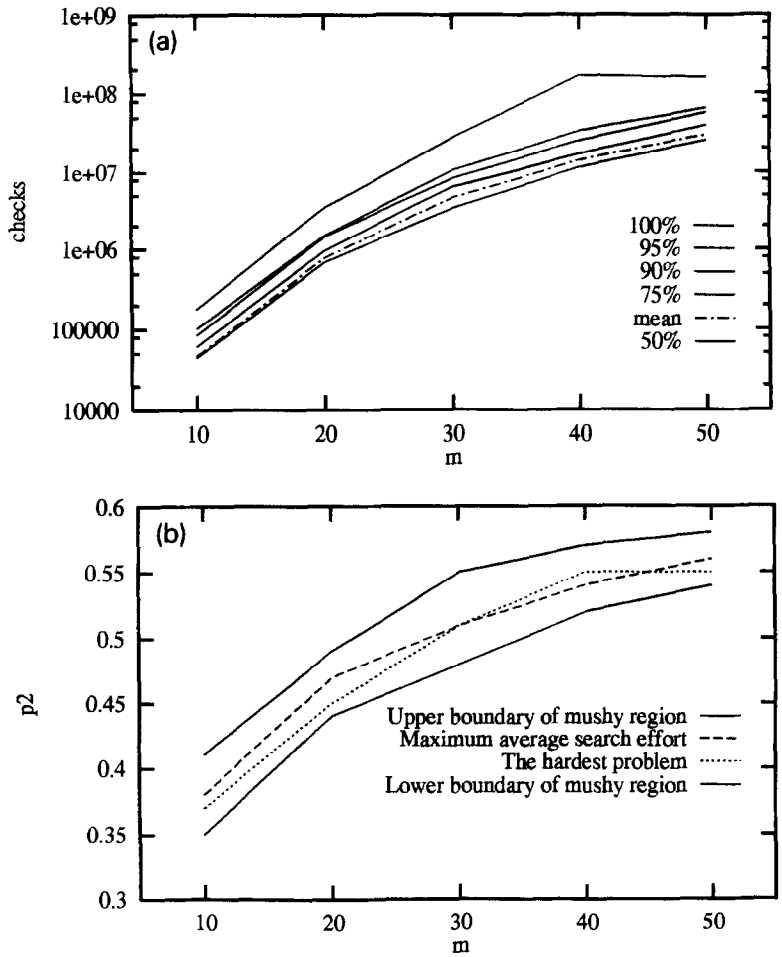

Fig. 12. Search effort against domain size, m, for $\langle 20, m, 0.1\rangle$ and locations of the complexity peaks and the boundaries of the mushy region.

where CBJ-MC found hard problems at values well below $p_{2\text{crit}}$, but the default algorithm found the exact same problems easy. For example, 1,000 problems were examined at $\langle 20, 10, 0.5, 0.27\rangle$. One problem was discovered where CBJ-MC took in excess of 640,000 checks, whereas the default algorithm solved the same problem in less than 3,000 checks. This problem was considered easy, as $p_{2\text{crit}}$ occurred at about 0.37 (see Fig. 1) and on average CBJ-MC solved the problems in thousands of consistency checks. This suggests that the phenomenon of exceptionally hard problems [12, 14, 21, 39] is to a significant extent algorithm-dependent.

Forward checking was used, with the fail-first heuristic (FC-FF), for a limited amount of experiments. It was observed that FC-FF performed worse than CBJ-MC at low values of p_2 for all problems. This was due to the relative abundance of solutions and the ineffectiveness of domain filtering in FC at low p_2. In very sparse constraint graphs

($p_1 = 0.1$) CBJ-MC was much better than FC-FF, and this was due to the difference between backjumping and chronological backtracking. It is a conjecture, that in order for an algorithm to perform well in sparse constraint graphs, it must be able to jump about the search space freely. When p_1 increased (beyond $p_1 = 0.3$) the difference in performance between FC-FF and the default algorithm became insignificant. It is a conjecture that this is due to the fail-first heuristic exposing dependencies; i.e. given a uniform domain size, when choosing the current variable fail-first will select a variable that has been filtered by some past variable, consequently when backtracking on failure the algorithm tends to fall back on a dependent variable.

Forward checking without the fail-first heuristic was almost always hopelessly inefficient. For example, the $\langle 20, 10, 0.1 \rangle$ experiments were re-run with 50 problems at each point (i.e. 4,950 problems), using the algorithms fc, FC-FF, FC-CBJ, and FC-CBJ-FF. Without the FF heuristic FC took 574 hours and FC-CBJ took 14 minutes. With the FF heuristic FC-FF took 6 minutes and FC-CBJ-FF took 2 minutes. The algorithms were encoded in C for these experiments [25]. However, there were particular cases where using the fail-first heuristic degraded performance. When constraints are loose (low values of p_2) it is in fact beneficial to choose as the current variable the one with most values remaining in its domain. Consequently when instantiating the current variable there are fewer variable/value pairs to check against in the future. Admittedly this may sound a bit artificial, but it suggests that when a problem is loosely constrained, constraint propagation may be wasteful (just check backwards), but if propagation is used then propagate from variables with large domains to variables with small domains. More information on the run time of the experiments is given in Appendix A.

5. Comparison with the theory

Williams and Hogg derived a theory that predicts that the phase transition occurs when there is a critical number of local nogoods per variable within a problem [44]. In the binary CSP a local nogood corresponds to a conflict between a pair of variables. The number of nogoods per variable is referred to as β and is calculated via Eq. (1), where n is the number of variables, m is the uniform domain size, p_1 is the density of the constraint graph, and p_2 is the tightness of the constraints. The critical number of nogoods per variable, β_{crit}, is given by Eq. (2) (and it should be noted that these equations result from applying the translations in Table 1 to the equations in [44]).

$$\beta = \tfrac{1}{2} p_1 (n-1) p_2 m^2, \tag{1}$$

$$\beta_{\text{crit}} = -\frac{p_2 m^2 \ln m}{\ln(1-p_2)}. \tag{2}$$

Consequently, the phase transition should occur when $\beta = \beta_{\text{crit}}$, and an estimated value of $p_{2\text{crit}}$, namely $\hat{p}_{2\text{crit}}$ (i.e. the expected value of constraint tightness where problems are on average most difficult), can then be derived:

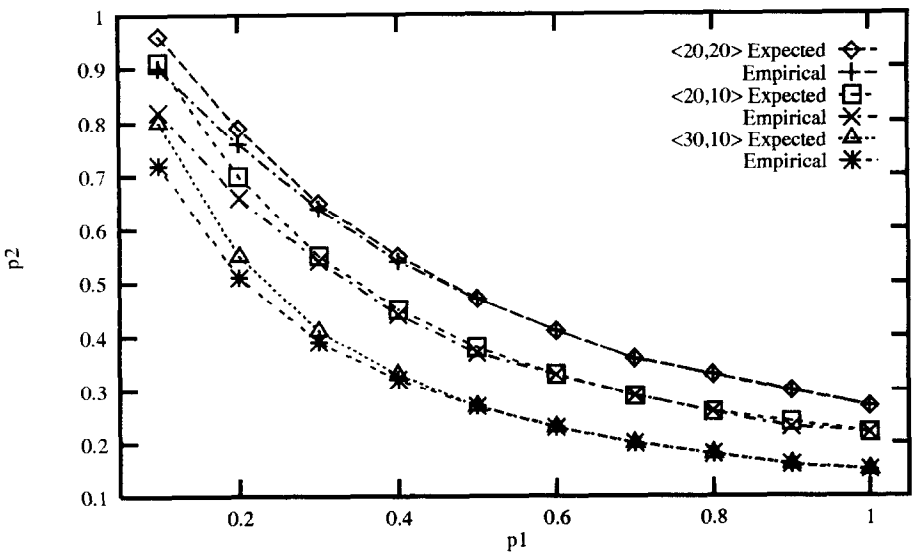

Fig. 13. Expected and empirical results.

$$\begin{aligned}
&\beta = \beta_{\text{crit}},\\
&\frac{1}{2}p_1(n-1)\hat{p}_{2\text{crit}}m^2 = -\frac{\hat{p}_{2\text{crit}}m^2 \ln m}{\ln(1-\hat{p}_{2\text{crit}})},\\
&\hat{p}_{2\text{crit}} = 1 - m^{-2/p_1(n-1)}.
\end{aligned} \tag{3}$$

It should be noted that $\hat{p}_{2\text{crit}}$ has been derived by different means. Smith [37,38] conjectures that the phase transition occurs when problems have, on average, just one solution. Given a binary CSP $\langle n, m, p_1, p_2\rangle$ an arbitrary instantiation of the variables will be a solution if all $\frac{1}{2}p_1 n(n-1)$ constraints are satisfied. There are m^n possible instantiations of the variable. Therefore, the expected number of solutions, $E(N)$, is:

$$E(N) = m^n(1-p_2)^{p_1 n(n-1)/2} \tag{4}$$

and the most difficult problems might occur when $E(N) = 1$, such that there is a mixture of insoluble problems and problems which have very few solutions, therefore

$$1 = m^n(1-\hat{p}_{2\text{crit}})^{p_1 n(n-1)/2}, \tag{5}$$

and this reduces to Eq. (3) above.

Theory and observation are presented in Fig. 13 for the $\langle 20, 10\rangle$, $\langle 30, 10\rangle$, and $\langle 20, 20\rangle$ experiments. The x-axis is p_1 and the y-axis is p_2, and the curves show where the observed and expected values of $p_{2\text{crit}}$ occur. There is a close match between the observed and expected values of $p_{2\text{crit}}$ at moderate and high values of p_1, but at low values of p_1

Table 3
The effect of being connected

Problem	p_{2crit}	$\hat{p}_{2crit}$	$p_{connected}$
$\langle 20, 10, 0.2 \rangle$	0.66	0.70	0.74
	0.67		1.00
$\langle 20, 10, 0.3 \rangle$	0.54	0.55	0.97
	0.52		1.00
$\langle 30, 10, 0.1 \rangle$	0.72	0.80	0.25
	0.72		1.00
$\langle 30, 10, 0.2 \rangle$	0.51	0.55	0.95
	0.52		1.00
$\langle 40, 10, 0.1 \rangle$	0.63	0.69	0.49
	0.63		1.00
$\langle 50, 10, 0.1 \rangle$	0.56	0.61	0.75
$\langle 60, 10, 0.1 \rangle$	0.50	0.54	0.87

there is a significant difference.[9] At low values of p_1 the hard problems occur earlier than expected. Experiments were performed to determine if this discrepancy might be due to the constraint graphs being disconnected. That is, if the graphs were disconnected they might appear to be made up of a smaller number of variables, and the actual density of the largest component may be greater than p_1. This would then require a smaller value of p_2 to become hard. The $\langle 20, 10, 0.2 \rangle$, $\langle 20, 10, 0.3 \rangle$, $\langle 30, 10, 0.1 \rangle$, $\langle 30, 10, 0.2 \rangle$, and $\langle 40, 10, 0.1 \rangle$ experiments were then repeated but any disconnected constraint graphs were discarded.[10] The results of those experiments are summarised in Table 3.

The column $p_{connected}$ is the ratio of connected constraint graphs over the number of constraint graphs examined. Where there are two row entries for a problem $\langle n, m, p_1 \rangle$, the first row corresponds to the experiments where constraint graphs were allowed to be disconnected, and the second row corresponds to experiments where disconnected graphs were discarded (consequently $p_{connected}$ will be 1.0). As n increases (and as p_1 increases) the probability that the constraint graph is connected increases, and this is expected [1,29]. Being connected does not appear to influence the outcome of the experiments; p_{2crit} and the location of the mushy region is generally unaffected. Although not shown, it did not significantly affect the average search effort either.

Fig. 14 shows the location of the theoretical and empirical results (peak in mean search effort) with respect to the mushy region for the $\langle n, 10, 0.1 \rangle$ experiments (i.e. n varying in sparse constraint graphs) and $\langle 20, m, 0.5 \rangle$ experiments (i.e. m varying in moderately dense constraint graphs).

In the sparse graphs, Fig. 14(a), we see that the theory consistently places $\hat{p}_2$ beyond the upper boundary of the mushy region, i.e. in the region where all problems are

[9] It is interesting to note that the theory agrees with Gaschnig's observations on randomly generated n-queens problems [11]. Gaschnig's problems corresponded to $\langle 10, 10, 1.0, p_2 \rangle$ and it was observed that there was a complexity peak at $L \approx 0.6$, where $L = 1 - p_2$. Theory predicts that p_{2crit} will be 0.40.

[10] The $\langle 20, 10, 0.1 \rangle$ experiments were not repeated, because all the connected $\langle 20, *, 0.1 \rangle$ problems are the path graphs P_n.

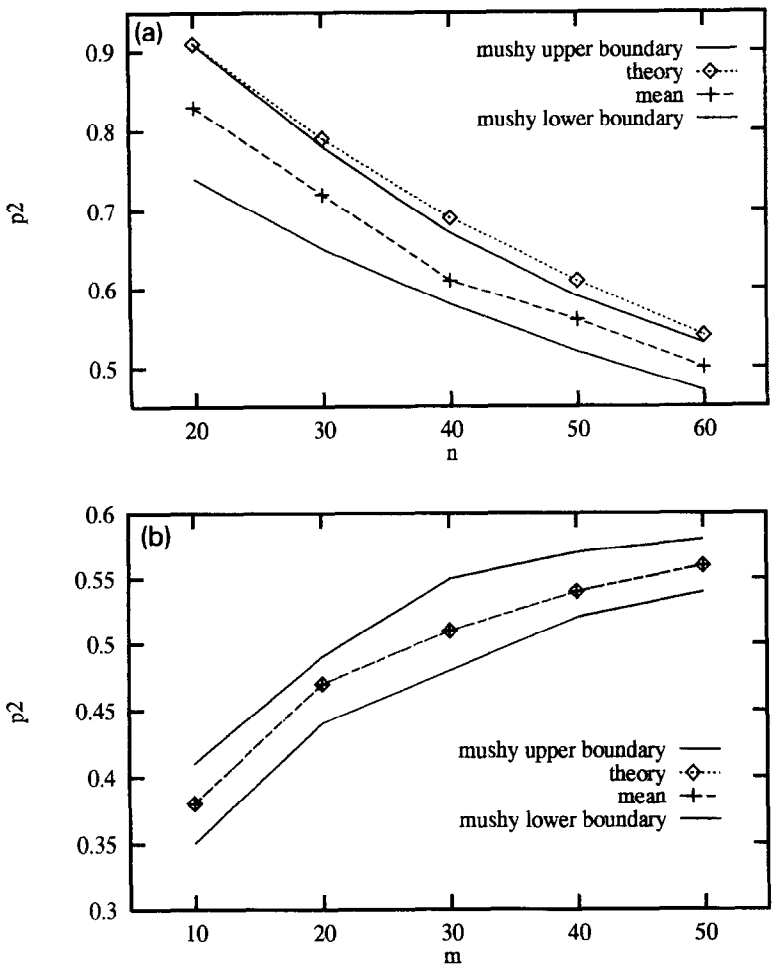

Fig. 14. Comparison of empirical and theoretical results, for (a) $\langle n, 10, 0.1\rangle$ and (b) $\langle 20, m, 0.5\rangle$.

insoluble. Conversely, in Fig. 14(b), theory matches the empirical results remarkably well; in fact, they are indistinguishable. One possible explanation for the discrepancy in Fig. 14(a) and the accuracy in Fig. 14(b) is as follows. The theory is based on the conjecture that on average the hardest problems will occur when on average there is a single solution. It may be the case that in sparse constraint graphs, at the crossover point problems either have very many solutions or none at all. Consequently, the average number of solutions is large at crossover, and it is only when constraints are tighter that problems have on average a single (or very few) solutions, and this will occur at the upper boundary of the mushy region. Conversely, in relatively dense constraint graphs, at the crossover point problems have either no solution or very few solutions. Consequently, theory becomes more accurate as density increases, and this agrees with Smith's observations on the variance in the number of solutions [38].

6. Conclusion

The study shows that for binary CSPs with n variables, uniform domain size m, and graph density p_1, there is a critical value of constraint tightness p_2, namely $p_{2\text{crit}}$, where the average search effort is maximal, and this correspond closely to the crossover point where 50% of the problems are soluble. The region where the phase transition takes place has been referred to as the mushy region, i.e. the region where problems change from being soluble to insoluble. As the density of the constraint graph increases the width of the mushy region decreases and the maximal average search effort increases.

The hardest problem was generally encountered on first entering the mushy region, and that problem was generally insoluble. Within the mushy region the insoluble problems were on average harder than the soluble problem. As p_2 increased the soluble problems became scarce and more difficult, and insoluble problems became plentiful and less difficult. These phenomena appear to be characteristics of the problem i.e. they are algorithm-independent. The landscape of the difficulty of some $\langle n, m \rangle$ problems have been mapped out and the topography is a ridge that curves from right to left, climbing as it goes.

It has been observed that in sparse constraint graphs (low values of p_1) there is great variability in search effort well before the phase transition. However, this variability was not enough to influence the location of the peak in the mean search effort, the mean occurring close to the peak in the median search effort. However, this might not be the case if larger samples of problems were examined, or less informed algorithms were used. As the density of the constraint graph increases this variability diminishes.

The location of the greatest average difficulty, $p_{2\text{crit}}$, was predicted to occur when an ensemble of problems have on average one solution. The theory was accurate for moderate to high values of p_1, but in sparse graphs theory predicted that $p_{2\text{crit}}$ would occur beyond the upper boundary of the mushy region, where all problems are insoluble. One possible explanation for this is that in the mushy region sparse problems have either no solution or very many solutions. Consequently, in sparse graphs at the crossover point the average number of solutions could be large, and it is only when nearly all problems are insoluble that the average number of solutions is low. Conversely, in moderate to high densities it appears that at the crossover point problems either have no solutions or very few solutions, and the theory is accurate.

The theory may be used in a number of ways. The most obvious role is within an empirical science of algorithms [22]. As a starting point, $p_{2\text{crit}}$ may be considered as a point of reference on the landscape, and algorithms may be positioned with respect to this. This might lead us to making informed choices as to what algorithm and heuristic to use for a given instance of a problem. Work in this area has already started [41]. Furthermore, by discovering under what circumstances one algorithm outperforms another we might gain a better understanding of the behaviour of algorithms, and this in turn should lead us to designs for even better algorithms and heuristics. Finally, this might lead us to a better understanding of the structure of hard problems.

Looking further ahead, if it is discovered that real world problems behave in a manner predicted by the theory then it might allow us to move intelligent decision making into the realms of real time. For example, if we have a resource allocation problem

Table A.1
Computational cost of the experiments

Experiment	#problems	Implementation	CPU time (hours)
⟨20, 10⟩	99,000	SCLisp 4.0	38
⟨20, 10, 0.5⟩	9,900	SCLisp 4.0	1.4
⟨20, 10, 0.5⟩	9,900	SCLisp 4.0 CBJ-MC	18
⟨20, 10, 0.5, 0.35⟩	1,000	SCLisp 4.0	1.2
⟨20, 10, 0.5, 0.36⟩	1,000	SCLisp 4.0	2.4
⟨20, 10, 0.5, 0.37⟩	1,000	SCLisp 4.0	3.5
⟨20, 10, 0.5, 0.38⟩	1,000	SCLisp 4.0	3.4
⟨20, 10, 1.0⟩	9,900	SCLisp 4.0	8
⟨20, 10, 1.0, 0.21⟩	1,000	SCLisp 4.0	13
⟨30, 10⟩	99,000	ANSI C	115
⟨30, 10, 0.1⟩	9,900	ANSI C	0.6
⟨40, 10, 0.1⟩	9,900	ANSI C	2
⟨50, 10, 0.1⟩	9,900	ANSI C	13
⟨60, 10, 0.1⟩	9,900	ANSI C	86
⟨20, 20⟩	99,000	ANSI C	188
⟨20, 20, 0.5⟩	9,900	SCLisp 4.0	90
⟨20, 20, 0.5⟩	9,900	ANSI C	11
⟨20, 30, 0.5⟩	9,900	ANSI C	55
⟨20, 40, 0.5⟩	9,900	ANSI C	135
⟨20, 50, 0.5⟩	9,900	ANSI C	359

characterised as $\langle n, m, p_1, p_2 \rangle$, in the event that $p_2 > \hat{p}_{2\text{crit}}$ we might know (with some confidence) that the problem is over-constrained. What is more relevant is that we might know how far the problem should be relaxed such that it becomes soluble, and possibly easy.

Appendix A. Computational effort

Table A.1 summarises the computational effort associated with the experiments. The first column describes the experiment, the second gives the number of problems examined, and the third column gives the implementation of the algorithm. If the implementation is SCLisp 4.0 then the default algorithm (FC-CBJ-DKC-FF) was encoded and run as compiled SUN Common Lisp version 4.0 on a SPARCstation IPC with 16MB of RAM. Up to three such workstations were used. If the implementation is ANSI C then the algorithms were encoded in ANSI C, compiled using *acc*, and the experiments were dispatched across 150 workstations. The majority of those workstations (110) were SPARCstation 1 and 1+ with 16MB of RAM, and 40 were SPARCstation ELCs with 16MB of RAM. The fourth column of Table A.1 gives the amount of hours of CPU time that was spent searching over the problems. That is, the amount of time to create the problems is not included in these figures.

Examining the ⟨20, 10, 0.5⟩ experiments it can be seen that the default algorithm is about an order of magnitude better than CBJ-MC (1.4 hours versus 18 hours). Looking

at the $\langle 20, 20, 0.5 \rangle$ experiments it can be seen that the ANSI C encoding was about an order of magnitude faster than the SCLisp encoding (11 hours against 90 hours). Furthermore, only three workstations were available with SCLisp compared to 150 with ANSI C. Consequently the C encoding allowed us to do about 500 times more experiments. In total, more than 1,100 hours of CPU time was used, and more than 400 thousand problems were examined. Table A.1 might serve two purposes. Firstly, if these experiments are repeated one will have an idea of the computational effort required. Secondly, as better algorithms and more computational resource becomes available we can look back and see how far we have progressed.

Acknowledgements

Ewan MacIntyre recoded the algorithms in C and wrote the dispatcher [25]. This allowed me to do the experiments 500 times quicker. The University of Strathclyde supplied the CPU cycles. I would like to thank Barbara Smith, Tad Hogg, and Ian Gent for all their help and encouragement, and my anonymous referee who suggested many important improvements to this paper.

References

[1] B. Bollobás, *Random Graphs* (Academic Press, New York, 1985).
[2] P. Cheeseman, B. Kanefsky and W.M. Taylor, Where the *really* hard problems are, in: *Proceedings IJCAI-91*, Sydney, Australia (1991) 331–337.
[3] J.M. Crawford and L.D. Auton, Experimental results on the crossover point in satisfiability problems, in: *Proceedings AAAI-93*, Washington, DC (1993) 21–27.
[4] R. Dechter, Enhancement schemes for constraint processing: backjumping, learning, and cutset decomposition, *Artif. Intell.* **41** (3) (1990) 273–312.
[5] R. Dechter, Constraint networks, in: *Encyclopedia of Artificial Intelligence* (Wiley, New York, 2nd ed., 1992) 276–286.
[6] R. Dechter and I. Meiri, Experimental evaluation of preprocessing algorithms for constraint satisfaction problems, *Artif. Intell.* **68** (2) (1994) 211–242.
[7] E.C. Freuder and R.J. Wallace, Partial constraint satisfaction, *Artif. Intell.* **58** (1-3) (1992) 21–70.
[8] D. Frost and R. Dechter, In search of the best search: an empirical evaluation, in: *Proceedings AAAI-94*, Seattle, WA (1994) 301–306.
[9] D. Frost and R. Dechter, Dead-end driven learning, in: *Proceedings AAAI-94*, Seattle, WA (1994) 294–300.
[10] J. Gaschnig, A general backtracking algorithm that eliminates most redundant tests, in: *Proceedings IJCAI-77*, Cambridge, MA (1977) 457.
[11] J. Gaschnig, Performance measurement and analysis of certain search algorithms, Tech. Rept. CMU-CS-79-124, Carnegie-Mellon University, Pittsburgh, PA (1979).
[12] I.P. Gent and T. Walsh, Easy problems are sometimes hard, *Artif. Intell.* **70** (1–2) (1994) 335–346.
[13] I.P. Gent and T. Walsh, The SAT phase transition, in: *Proceedings ECAI-94*, Amsterdam, Netherlands (1994) 105–109.
[14] I.P. Gent and T. Walsh, Computational phase transitions in real problems, Research Paper 724, Department of Artificial Intelligence, Edinburgh University, Scotland (1994).
[15] I.P. Gent and T. Walsh, The TSP phase transition, Technical Report, Department of Computer Science, University of Strathclyde, Scotland (1995).
[16] M.L. Ginsberg, Dynamic backtracking, *J. Artif. Intell. Res.* **1** (1993) 25–46.

[17] M.L. Ginsberg, M. Frank, M. Halpin, M.P. Halpin and M.C. Torrance, Search lessons learned from crossword puzzles, in: *Proceedings AAAI-90*, Boston, MA (1990) 210–215.
[18] S.W. Golomb and L.D. Baumert, Backtrack programming, *J. ACM* **12** (1965) 516–524.
[19] R.M. Haralick and G.L. Elliott, Increasing tree search efficiency for constraint satisfaction problems, *Artif. Intell.* **14** (1980) 263–313.
[20] A. Haselböck, Exploiting interchangeabilities in constraint satisfaction problems, in: *Proceedings IJCAI-93*, Chambery, France (1993) 282–287.
[21] T. Hogg and C.P. Williams, The hardest constraint problems: a double phase transition, *Artif. Intell.* **69** (1-2) (1994) 359–378.
[22] J.N. Hooker, Needed: An empirical science of algorithms, *Operations Research* **42** (2) (1994) 201–212.
[23] S. Kirkpatrick and B. Selman, Critical behavior in the satisfiability of random boolean expressions, *Science* **264** (1994) 1297–1301.
[24] V. Kumar, Algorithms for constraint satisfaction problems: a survey, *AI Mag.* **13** (1) (1992) 32–44.
[25] E. MacIntyre, Really hard problems, Final Year Report for the B.Sc. Degree in Computer Science, Department of Computer Science, University of Strathclyde, Scotland (1994).
[26] A.K. Mackworth, Constraint satisfaction, in: *Encyclopedia of Artificial Intelligence, Vol.* 1 (Wiley, New York, 2nd ed., 1992) 285–293.
[27] P. Meseguer, Constraint satisfaction problems: an overview, *AI Comm.* **2** (1) (1989) 3–17.
[28] D. Mitchell, B. Selman and H.J. Levesque, Hard and easy distribution of SAT problems, in: *Proceedings AAAI-92*, San Jose, CA (1992) 459–465.
[29] E.M. Palmer, *Graphical Evolution* (Wiley, New York, 1985).
[30] P. Prosser, Hybrid algorithms for the constraint satisfaction problem, *Comput. Intell.* **9** (3) (1993) 268–299.
[31] P. Prosser, Domain filtering can degrade intelligent backtracking search, in: *Proceedings IJCAI-93*, Chambery, France (1993) 262–267.
[32] P. Prosser, BM + BJ = BMJ, in: *Proceedings CAIA-93*, Orlando, FL (1993) 257–262.
[33] P. Prosser, Binary constraint satisfaction problems: some are harder than others, in: *Proceedings ECAI-94*, Amsterdam, Netherlands (1994) 95–99.
[34] P. Prosser, A csp lab, *ftp site* ftp.cs.strath.ac.uk, *username* anonymous, *password* your full email address, *directory* local/pat/csp-lab (1995).
[35] P.W. Purdom, Search rearrangement backtracking and polynomial average time, *Artif. Intell.* **21** (1983) 117–133.
[36] D. Sabin and E.C. Freuder, Contradicting conventional wisdom in constraint satisfaction, in: *Proceedings ECAI-94*, Amsterdam, The Netherlands (1994) 125–129; also in: *Proceedings AAAI-92*, San Jose, CA (1992) 440–446.
[37] B.M. Smith, Phase transition and the mushy region in constraint satisfaction problems, in: *Proceedings ECAI-94*, Amsterdam, Netherlands (1994) 100–104.
[38] B.M. Smith and M.E. Dyer, Locating the phase transition in binary constraint satisfaction problems, *Artif. Intell.* **81** (1996) 155–181 (this volume).
[39] B.M. Smith and S.A. Grant, Sparse constraint graphs and exceptionally hard problems, Research Report 94.36, Division of Artificial Intelligence, School of Computer Studies, University of Leeds, England (1994).
[40] E.P.K. Tsang, *Foundations of Constraint Satisfaction* (Academic Press, New York, 1993).
[41] E.P.K Tsang, J. Borrett and A.C.M Kwan, An attempt to map the performance of a range of algorithm and heuristic combinations, in: *Proceedings AISB-95* (1995).
[42] C.P. Williams and T. Hogg, Extending deep structure, in: *Proceedings AAAI-93*, Washington, DC (1993).
[43] C.P. Williams and T. Hogg, The typicality of phase transitions in search, *Comput. Intell.* **9** (3) (1993) 211–238.
[44] C.P. Williams and T. Hogg, Exploiting the deep structure of constraint problems, *Artif. Intell.* **70** (1–2) (1994) 73–118.