# Phase Transitions and Coarse-Grained Search

Colin P. Williams and Tad Hogg
Xerox Palo Alto Research Center
Palo Alto, CA 94304
CWilliams@parc.xerox.com, hogg@parc.xerox.com

## Abstract

*Abstraction is a method for solving a variety of computational search problems that uses coarse-graining to simplify the search. When a coarse-grained, or abstract, solution is found, it is then refined to give a complete solution. We present a model of this abstraction process for constraint satisfaction problems, a well-known class of NP-complete search problems. This model is then used to identify phase-transition like behavior in the effectiveness of abstraction as well as determine the type of abstraction likely to be most useful for relatively hard instances of these search problems.*

## 1: Introduction

Many computational problems can be couched as a search through some space of possibilities. Unfortunately, the number of such possibilities often grows exponentially or worse with the size of the problem. In such circumstances it is infeasible to search through each possibility explicitly. Instead computer scientists have invented ways of guiding the search in some principled fashion. One of these techniques, called "abstraction", consists of mapping the given problem into a simpler problem, solving the simpler problem and then inverting the mapping to obtain guidance as to how to solve the original problem. For example, when planning a trip one can first solve a simplified, or abstracted, version of the problem that considers only the possible flights between various airports. Once that problem is solved, its solution can be used as a guide to fill in the details, e.g., determining how to get to and from the airports. This illustrates the important aspects of abstraction: to be useful, it must be relatively easy to obtain a solution to the abstract problem and it must also be possible, without excessive search, to use the abstract solution to construct a full solution to the problem. Abstraction is now one of the most active areas of computer science and artificial intelligence research [6, 7, 10, 11, 14, 1].

Unfortunately, progress in understanding the nature of abstraction has been slow. Most work to date relies on logical analyses [6, 7] which generally provide insight into questions relating to the properties that are or are not retained under alternate abstraction mappings but they are not suited to understanding the dynamics of search with abstraction.

A more productive point of view arises from the observation that large computational problems involve many interacting components whose detailed configuration is unspecified or changing [9]. Moreover, search algorithms are desired that work reasonably well over an ensemble of problems rather than only a few special instances (i.e., the methods should not be brittle). Both these observations suggest that most of the detailed structure of large computational search problems is either not known *a priori* or cannot be relied upon by robust algorithms. Hence we are led to consider classes of problems with many unspecified or "internal" degrees of freedom. In such cases it is useful to treat these internal degrees of freedom as randomly generated by some probability distribution. In this setting the main issue is then to relate the overall or global behavior of a search method to a few parameters describing the problem ensemble and search method. To be readily measurable, these parameters will generally relate to local properties of the problems or algorithms. Thus one might expect that statistical techniques, which have been so successful in describing physical systems, will provide a useful framework for understanding the global behavior of these computational problems, particularly when moving beyond idiosyncratic small systems. This approach is likely to provide insight into the generic phenomena of large software systems, beyond the just the limits imposed by the physical nature of hardware [8].

In this paper we use these insights to present an alternate view of the abstraction process. Specifically, abstraction can be viewed as a type of "coarse graining" familiar from theoretical physics. This model provides a more comprehensive picture of abstraction and predicts certain computational phenomena that are not apparent

| Parameter | Meaning |
|---|---|
| $\mu$ | number of variables |
| $b$ | number of values per variable |
| $m$ | number of minimized nogoods |
| $k$ | average size of minimized nogoods |

**Fig. 1. A coarse description of a CSP.**

from the more traditional "logical" analyses.

## 2: A Universal Model of Constraint Satisfaction Problems

To convey some sense of the generality of our results we will frame our discussion in the context of "constraint satisfaction problems" (CSPs). We chose to think of CSPs that could be represented as a set of constraints over $\mu$ variables, each of which can take on one of $b$ values. Each constraint determines whether a particular combination of assignments of values to the variables are consistent ("good") or inconsistent ("nogood"). Collecting the nogoods of all the constraints and discarding any that are supersets of any other we arrive at a set of $m$ "minimized nogoods" which completely characterize the particular CSP. In general these nogoods can involve different numbers of variables but for simplicity we'll assume they are all of size $k$. Thus in our model, a CSP can specified using just 4 parameters, $\{\mu, b, m, k\}$. On average any such tuple will admit $\langle N_{soln} \rangle$ solutions (a solution being a set of assignments of values to variables such that all the constraints are satisfied simultaneously) where $\ln \langle N_{soln} \rangle \sim \mu[\ln(b) + \beta \ln(1 - b^{-k})]$ (see [15, 16]).

A concrete example is provided by the graph colouring CSP. This consists of a graph containing $\mu$ nodes (i.e. variables) that each have to be assigned one of $b$ colours (i.e. values) such that no two nodes at either end of an edge have the same colour (i.e. such that no constraints are violated). As there are $b$ possible colours, each edge introduces exactly $b$ nogoods each of size $k = 2$. Consequently, in total, there are $m = eb$ nogoods where $e$ is the number of edges in the graph.

What is interesting is that this coarse level of description turns out to be sufficient to predict parameter regimes in which CSP instances will be especially costly to solve.

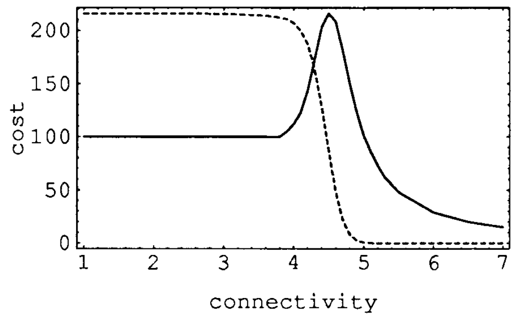

**Fig. 2. Behaviour for 3–colouring of random graphs with 100 nodes as a function of connectivity $\gamma$ in steps of 0.1. The solid curve shows the median search cost, and the dashed one is the fraction of graphs with a solution (ranging from one on the left to zero on the right).**

## 3: Phase Transitions & Problem Structure

"Computational cost" has traditionally been taken to be the number of elementary search steps that some algorithm must take in order to solve a particular problem or to prove it insoluble. At one time people were content to estimate best, worst or average cost across a sample set consisting of all instances of problems of a given size. For example, if one were considering the problem of colouring a graph consisting of $n$ nodes with at most $b$ colours then the different problem instances would correspond to different possible choices of the number and placement of the edges. More recently, attention has shifted to a finer grain analysis. Instead of lumping all graphs together the new experiments partition the sample into equivalence classes consisting of graphs with equal numbers of edges, $e$ or equivalently equal connectivity[1], $\gamma = \frac{2e}{n}$.

If the median cost for solving the sample of problems in each equivalence class is plotted as a function of the average connectivity, $\gamma$, (see Figure 2) a sharp peak is revealed at a particular connectivity which, for the case of 3–colouring, occurs around $\gamma = 4.6$ [3, 15]. This peak appears to coincide with the transition from underconstrained to overconstrained problems, as evidenced by a step in the probability that a problem instance has a solution. Moreover if the experiment is repeated with larger sized problems (i.e. bigger $n$'s) the location of the peak remains fixed but it becomes even sharper and taller. More surprising still, if the experiments are repeated with *different* graph colouring

[1] Connectivity is a quantity that can be kept invariant as the size of the graphs under investigation is increased thereby allowing more meaningful comparisons to be made between graphs containing different numbers of nodes.

algorithms the location of the peak remains more or less the same although the absolute values of the search costs, in particular the peak costs, changes. This suggests that the location of the peak in the median has more to do with the intrinsic structure of the problems rather than the idiosyncracies of particular graph colouring algorithms.

A similar study of $k$-SAT [2, 13, 15, 16, 12, 5], a different NP-complete problem, for which the structural parameter is the ratio of the number of clauses to variables, reveals a similar easy-hard-easy pattern.

## 4: Relating Abstraction to the Phase Transition

Whatever the precise details, all methods of abstraction begin by mapping the ground problem into some other problem. In terms of our model we can therefore think of an abstraction as a mapping between two representations:

$$\{\mu, b, m, k\} \to \{\mu', b', m', k'\} \quad (1)$$

with a corresponding change in the expected number of solutions from $N_{soln} \to N'_{soln}$. Then a "solution" in the abstract space is mapped back to the ground space to guide the search for a solution to the original problem.

Three key notion characterize an abstraction: its **reliability** (the probability $p$ with which each step in the search in the abstract space induces the correct steps in the ground space when it is mapped back), its **strength**, $\frac{\mu}{\mu'}$ (the ratio of the size of a solution in the ground space to that in the abstract space) and its **provability**, $\alpha = N'_{soln}$ (the number of solutions or "proofs" in the abstract space).

Since abstraction maps a given search problem to a smaller one, this in effect produces a "coarse graining" of the original problem. This can take place in a variety of ways. A simple example is when several variables are grouped together into a single macrovariable. Alternatively, the possible values can be grouped together, e.g., into various ranges, and finally the constraints can be weakened or strengthened. In all cases, abstraction produces a new, hopefully simpler, search problem whose solutions may give a coarse representation of one or more solutions to the original problem.

As the critical value of the structural parameter depends upon the variables mentioned in Eqn. 1 different abstraction mappings can not only lower search costs by reducing the *size* of the search space (i.e. by lowering $\mu$ and/or $b$) but also by *shifting* a problem farther from the "hard" region. Elsewhere we describe how particular abstraction mappings translate into explicit maps. But for

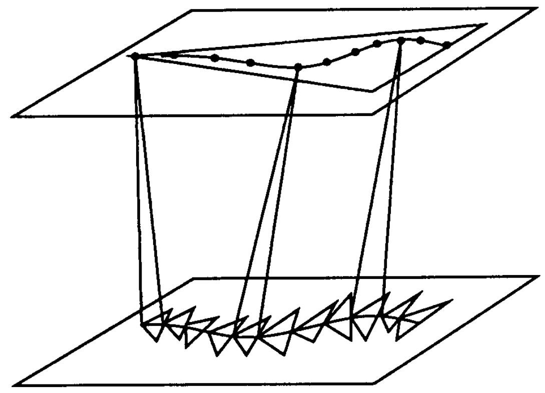

**Fig. 3. Characterization of constraint satisfaction with 1 level of abstraction. The steps in the abstract search define islands through which the search in the ground space is focussed.**

the purposes of this paper we will now introduce a simple phenomenological model of the *effects* of abstraction on the ground and abstract search spaces.

## 5: Phenomenological Model of Abstraction

In previous analyses [7, 10] it has always been assumed that the abstraction is perfectly reliable (i.e. $p = 1$) in the sense that once an abstract solution is found, a ground solution can be induced by expanding each step in the abstract search as a sequence of small search trees (each of size $T$ say) rooted at the image of one abstract node and terminating at the image of the adjacent node in the abstract search[2] (see Figure 3). Indeed, these studies also showed that the net cost was minimized when the "gaps" in the ground search were of equal size. In this case each of the little tree searches induced along the image of some abstract solution path can be crudely estimated to be of size $T = b^{\mu/\mu'}$.

We extend this work in a realistic direction by relaxing the assumption that the "mapping back" is perfect. This allows us to estimate the net search cost using one level of abstraction as a function of the *reliability* of the abstraction. We discover that there is a regime in which the net cost of constraint satisfaction with (imperfect) abstraction is worse than direct ground space search and that there is a critical reliability at which the net search cost, when using abstraction, suddenly drops abruptly. This

[2] That the large original ground tree can be replaced with a path of smaller search trees is precisely the reason why abstraction can be advantageous.

| Parameter | Interpretation |
|---|---|
| $p$ | **Reliability** of the Abstraction: probability with which each step in the abstract proof can be successfully expanded in the ground space |
| $\mu/\mu'$ | **Strength** of the abstraction: ratio of the length of a proof in the ground space to that in the abstract space |
| $T = b^{\mu/\mu'}$ | average size of the search tree generated for each step of an abstract proof expanded in the ground space |
| $\alpha = N'_{soln}$ | **Provability** of the Abstraction: number of "proofs" in the abstract space |
| $1 \leq \langle\Lambda\rangle \leq \alpha$ | the average trial, in a successful use of abstraction, at which an abstract proof first "goes through" in the ground space |
| $1 \leq \langle\lambda\rangle \leq \mu'$ | the average number of steps, in a failing trial, at which the ground proof actually fails |

**Fig. 4. Definitions of parameters in the model.**

suggests that, in the vicinity of the critical reliability, a small improvement could significantly reduce the cost.

To make the model more quantitative, we take the net cost of performing search with abstraction to be the sum of

1. the cost of making the abstraction(s)
2. the cost of search in the abstraction level(s)
3. the cost of making the unabstraction(s)
4. the cost of search in the ground level

As a first approximation we shall assume that the cost of the abstraction and unabstraction processes can be neglected compared to the costs of search in the ground and abstract spaces.

Assume there is a single solution path to be found in the ground space, corresponding to the original problem being roughly in the "hard" region ($N_{soln} = 1$), and $N'_{soln} = \alpha$ solution paths in the abstract space. As an abstraction mapping may be imperfect, we cannot guarantee that an abstract solution path will necessarily guide the ground solution path in the correct manner. Consequently, there is a chance that every abstract solution path fails to induce the right ground solution path. In this case, one must eventually resort to exhaustive search in the ground space in order to find the ground solution. The net cost of performing a search for a ground solution should therefore reflect the relative likelihood of success and failure of the abstraction procedure. Hence, we write the net cost as:

$$C = \left(C^{grd}_{succ} + C^{abs}_{succ}\right)p_{succ} + \left(C^{grd}_{fail} + C^{abs}_{fail}\right)(1 - p_{succ}) \tag{2}$$

where $p_{succ}$ is the probability that the search in the abstract space will ultimately guide the search in the ground space to a correct solution, $C_{succ}$ is the average cost of such a (2–level) search and $C_{fail}$ is the average cost of an unsuccessful search i.e. one that ultimately resorts to direct exploration of the ground space. Each of these costs can be partitioned into contributions arising from the search in the ground space ($C^{grd}_{succ}, C^{grd}_{fail}$) and contributions arising from search in the abstract space ($C^{abs}_{succ}, C^{abs}_{fail}$). Due to the assumption in our model that there *is* a solution to be found in the ground space, all searches ultimately result in finding it. The "succ" subscripts refer to the case when the ground solution was found because an abstract solution focussed attention along the right path. Similarly, the "fail" subscript indicates that the ground solutions was only found after all abstract solution paths had lead to dead ends and the search returned to exhaustive exploration of the ground space.

## Net Cost

By a similar analysis to that described in [14] we show that this cost can be approximated as:

$$\begin{aligned} C = &\left[\left(\left(\langle\Lambda\rangle - 1\right)\langle\lambda\rangle + \mu'\right)T + \frac{\langle\Lambda\rangle}{\alpha}\frac{b^{\mu'+1} - 1}{b - 1}\right]\left(1 - \left(1 - p^{\mu'}\right)^{\alpha}\right) \\ &+ \left[\alpha\langle\lambda\rangle T + \frac{b^{\mu'+1} - 1}{b - 1} + \frac{b^{\mu} - \alpha}{2b^{\mu}}\frac{b^{\mu+1} - 1}{b - 1}\right]\left(1 - p^{\mu'}\right)^{\alpha} \end{aligned} \tag{3}$$

where $p_{succ} = 1 - \left(1 - p^{\mu'}\right)^{\alpha}$, which resembles a step function, is the probability of at least one of the abstract solution paths inducing a successful ground search, $\langle\Lambda\rangle \approx \frac{1-\left(1-p^{\mu'}\right)^{\alpha}}{p^{\mu'}}$ is the average trial at which an abstract solution path *first* goes through all the way in the ground space and $\langle\lambda\rangle \approx \frac{1-p^{\mu'}}{1-p}$ is the average number of steps in a failing trial at which the trial fails.

## 6: Consequences of the Model

Analysis of this solution reveals the following computational phenomena:

1. there is a critical threshold reliability above which abstraction is beneficial and below which it is not (see Fig 5)

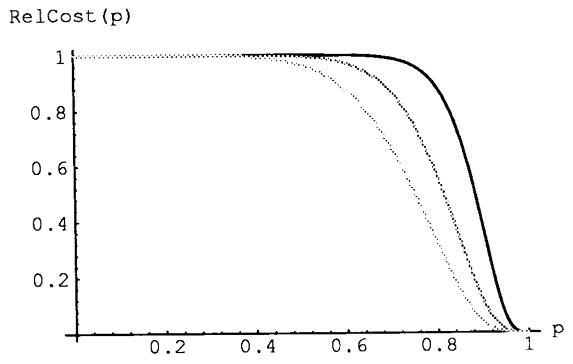

**Fig. 5. Relative cost of constraint satisfaction with abstraction to that without as a function of the reliability of the abstraction, $p$ for three values of the strength $\frac{\mu}{\mu'} = 3, \frac{\mu}{\mu'} = 5, \frac{\mu}{\mu'} = 7$ (dark to light). Notice the abrupt drop in the relative cost above a critical value of reliability.**

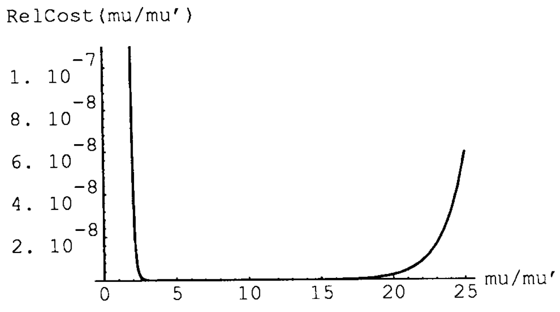

**Fig. 6. Relative cost of search with abstraction compared to that without, as a function of the strength of the abstraction, $\frac{\mu}{\mu'}$ for $\alpha = 1, b = 2, \mu = 50, p = 1$. Note that this case corresponds to a perfectly reliable abstraction (because $p = 1$) and confirms that there is an optimal strength for the abstraction at $\frac{\mu}{\mu'} = 6.5$. Similar results hold when $p < 1$.**

2. there is an optimal strength, $\frac{\mu}{\mu'}$ to an abstraction (for both the case of perfect and imperfect abstraction — see Fig. 6)
3. abstraction can work not only by reducing the size of the search space but also by shifting a problem in a hard region to an easier region.
4. abstraction has the most pronounced effect for problems (in the ground space representation) that are at the phase transition point.

## 7: Conclusions

We have described how a statistical model of the abstraction mapping for search provides insight into a variety of computational phenomena. These include the trade off between reliability and strength of the abstraction and the observation that abstraction could be improved, on average, by mappings that move away from the phase transition region with its concentration of relatively hard problems. Since the hard cases appear to arise from critically constrained problems, by weakening or strengthening some constraints, the resulting abstract problem will no longer be critically constrained and hence is likely to be easier to solve. Counterbalancing this is the question of reliability. However, even if the abstract solution does not expand to a complete solution to the original problem, it could still be useful in obtaining fairly good partial solutions.

These observations also suggest a further refinement. In many CSPs with local constraints, some variables will appear in more constraints than average, while others are in fewer. Combined with the general theory of CSP hardness described above, this could form the basis for abstractions that selectively strengthen or weaken constraints involved in different parts of the search space based on the likely hardness of the subproblems defined by those parts. Although it remains to be seen how well such a method would work in practice, this approach has provided a way to improve a different search method, genetic algorithms, in some cases [4].

## References

[1] F. Bacchus and Qiang Yang. Downward refinement and the efficiency of hierarchical problem solving. Technical Report CS-92-45, Dept. of Computer Science, Univ. of Waterloo, Ontario, Canada, 1992.

[2] Peter Cheeseman, Bob Kanefsky, and William M. Taylor. Where the really hard problems are. In J. Mylopoulos and R. Reiter, editors, *Proceedings of IJCAI91*, pages 331–337, San Mateo, CA, 1991. Morgan Kaufmann.

[3] Peter Cheeseman, Bob Kanefsky, and William M. Taylor. Computational complexity and phase transitions. In *Proc. of the Physics of Computation Workshop*. IEEE Computer Society, October 2-4 1992.

[4] Scott H. Clearwater and Tad Hogg. Exploiting problem structure in genetic algorithms. In *Proc. of the 12th Natl. Conf. on Artificial Intelligence (AAAI94)*, pages 1310–1315, Menlo Park, CA, 1994. AAAI Press.

[5] James M. Crawford and Larry D. Auton. Experimental results on the cross-over point in satisfiability problems. In *Proc. of the 11th Natl. Conf. on Artificial Intelligence (AAAI93)*, pages 21–27, Menlo Park, CA, 1993. AAAI Press.

[6] F. Giunchiglia and T. Walsh. Abstract theorem proving. In *11th IJCAI*, 1989.

[7] F. Giunchiglia and T. Walsh. Using abstraction. Technical Report IRST Tech. Rep. 9010-08, Istituto per la Ricerca Scientifica, Trento, Italy, 1990.

[8] T. Hogg and B. A. Huberman. Artificial intelligence and large scale computation: A physics perspective. *Physics Reports*, 156:227–310, 1987.

[9] J. O. Kephart, T. Hogg, and B. A. Huberman. Dynamics of computational ecosystems. *Physical Review A*, 40:404–421, 1989.

[10] C. A. Knoblock. Search reduction in hierarchical problem solving. In *AAAI-91*, pages 686–691, 1991.

[11] C. A. Knoblock and J. D. Tenenberg. Characterizing abstraction hierarchies for planning. In *AAAI-91*, pages 692–697, 1991.

[12] Tracy Larrabee and Yumi Tsuji. Evidence for a satisfiability threshold for random 3CNF formulas. In Haym Hirsh et al., editors, *AAAI Spring Symposium on AI and NP-Hard Problems*, pages 112–118. AAAI, 1993.

[13] David Mitchell, Bart Selman, and Hector Levesque. Hard and easy distributions of SAT problems. In *Proc. of the 10th Natl. Conf. on Artificial Intelligence (AAAI92)*, pages 459–465, Menlo Park, 1992. AAAI Press.

[14] C. P. Williams. Imperfect abstraction. In *AAAI Workshop on Abstraction and Approximation*, 1992.

[15] Colin P. Williams and Tad Hogg. Using deep structure to locate hard problems. In *Proc. of the 10th Natl. Conf. on Artificial Intelligence (AAAI92)*, pages 472–477, Menlo Park, CA, 1992. AAAI Press.

[16] Colin P. Williams and Tad Hogg. Extending deep structure. In *Proc. of the 11th Natl. Conf. on Artificial Intelligence (AAAI93)*, pages 152–157, Menlo Park, CA, 1993. AAAI Press.