Complexity-Theoretic Models of Phase Transitions in Search Problems*

Paul E. Dunne, Alan Gibbons, and Michele Zito
Department of Computer Science
University of Liverpool
Liverpool L69 7ZF
United Kingdom
{ped,amg,michele}@csc.liv.ac.uk

Abstract

In recent years, numerous studies have observed that many hard combinatorial decision problems exhibit behaviour described as a '*phase-transition*'. This is the phenomenon whereby typical instances of a problem display a dramatic shift in certain characteristics as some parameter of the instances is varied. Such characteristics include the likelihood of an instance having a solution and the time taken by a search algorithm. The apparent pervasiveness of phase-transitions in hard combinatorial search problems has led to contrasting claims being advanced concerning to what extent all *NP*-complete problems exhibit phase-transitions. The established importance of exploiting phase-transition effects in the design of search heuristics provides a strong motivation for assessing how valid such claims may be. In this paper we argue that questions concerning the generality of phase-transition phenomena are, at present, ill-defined. In order to address this difficulty, we propose and examine rigorous complexity-theoretic models of the statement 'the decision problem D has a phase-transition'. Within these models it is proved that for certain 'natural' definitions contrasting results about phase-transition behaviour can be proved.

Keywords: Computational Complexity, Phase-Transitions, Search Problems.

1. Introduction

Recent empirical studies on the performance of standard search heuristics applied to certain *NP*-complete problems (such as 3-*SAT*, 3-colouring, *TSP*) have noted a common property of these: that of having a *phase-transition*, e.g. (Cheeseman, Kanefsky, & Taylor, 1991), (Mitchell, Selman, & Levesque, 1992), (Hogg and Williams, 1994), etc. This property may be described, in loose terms, as an extreme change in the behaviour of some aspect of a problem that occurs when small changes are made to parameters of typical input instances. As a concrete example consider the decision problem 3-*SAT*. Suppose instances of 3-*SAT* are drawn, uniformly at random, from the set of formulae with n propositional variables and m clauses. Solving such instances with a heuristic such as that of (Davis and Putnam, 1960) reveals two patterns as the ratio m/n increases: the proportion of satisfiable instances moves from 'close to' one to 'close to' zero; the time taken by the search process changes from being 'low' to 'high' and back to 'low'. Furthermore, the transition from 'typical' instances being satisfiable to 'typical' instances being unsatisfiable is characterised by a sharp threshold value for m/n, experimentally revealed as around 4.2. The hardest instances, i.e. those which take longest to search, are also clustered in this region of clause-to-variable ratio. Similar behaviours, with appropriate parameters, have been observed for 3-colourability, Traveling Salesman, and a number of other *NP*-complete problems. It has been noted, see e.g. (Hogg, Huberman, & Williams, 1996), that such effects persist over a range of different search algorithms, leading a number of authors to moot the use of such transition thresholds as a basis for selecting or designing good search heuristics specialised to individual problems, e.g. the 'minimise κ' heuristic of (Gent *et al.*, 1996).

*) The work reported in this paper was carried out with support from EPSRC grant GR/L/77089

Our aim in this paper is to propose *analytic* frameworks, that are suited to investigating phase-transition phenomena as a *general* property of combinatorial decision problems. In particular, we propose a number of precise formulations of a conjecture that is implicit in several empirical studies and concerns the prevalence of phase-transition effects amongst *NP*-complete problems. Thus,

> "Such phase transition phenomena appear to be ubiquitous. Indeed we have yet to find an *NP*-complete problem which lacks a phase transition".
> (Gent and Walsh, 1996)

In contrast, other studies have advanced claims that particular *NP*-complete problems do *not* exhibit phase-transitions, e.g. (Kirkpatrick and Selman, 1994)

At present, the claims implied by such statements are incapable of resolution. This may not (necessarily) be for reasons of mathematical intractability but rather because their precise terms of reference are ill-defined. For example, what rôle is played by the *representation* of a given decision problem, i.e. the encoding of problem instances? What behaviour qualifies as a 'phase-transition'? What parameters of a problem instance is it 'reasonable' to measure in attempting to demonstrate a phase-transition? We first illustrate the non-trivial nature of these questions by analysing some possible definitions of the statement 'the decision problem D has a phase-transition'. A central contention of our formalism is that the manner in which a given decision problem exhibits a phase-transition is a combinatorial and complexity-theoretic issue and that, therefore, the natural mathematical language to adopt in defining such effects should be in terms of the framework of Computational Complexity Theory.

We note, in passing, that although (Gent *et al.*) attempt to study phase-transition phenomena in a general context, there are a number of weaknesses in their approach: the concept of 'ensemble', which is central to their model, is never precisely formulated; the type of phase-transition being described concerns the single characteristic of 'probability that an ensemble member has a solution' (the association with search cost is conjectural and supported through empirical study); and, finally, the formalism only provides an upper bound on the region where a phase-transition of this type occurs. As such, attempts to establish that some decision problems defined by their complexity classes exhibit phase-transitions through the sole basis of 'constrainedness of ensembles' would seem to be ill-fated.

One consequence of our approach is that a clear distinction is made between characteristics such as 'the probability of a typical instance having a solution' and characteristics such as 'the average or worst-case search time taken to evaluate an instance'. Analytic studies of phase-transitions have largely concentrated on the former, e.g. (Chvátal and Reed, 1992), (Koršunov, 1976), (Kirousis, Kranakis, and Krizanc, 1996), whereas most empirical studies have, in addition, examined phase-transitions in the performance of search methods. Within these models it turns out that using quite 'natural' parameters a number of general results about the presence or absence of specific phase-transition effects can be proved.

2. Defining Phase-Transitions

In this section we discuss the principal formal components that, we contend, are necessary to render meaningful, statements to the effect that 'f has/does not have a phase-transition'. In proposing these, our main aim is to substantiate the claims concerning why these particular elements are necessary, avoiding notational complexities as far as is consistent with preventing ambiguity. In order to discuss the characteristics of problems that are to be considered in examining phase-transitions we employ the following terminology.

Definition 1: A *decision problem* is a function $f : I \rightarrow \{ 0, 1 \}$ where I is called the set of *instances*. It is assumed that elements of a set of instances I are *represented* as finite strings over a finite *alphabet*, Σ, containing at least two symbols. We use $|x|$ to denote the number of symbols in x and, if $\Sigma = \{ 0, 1 \}$, $H(x)$ to denote the number of 1s in x. •

For example, suppose $I = \{ \textit{Undirected graphs} \}$. Letting $\Sigma = \{ 0, 1 \}$ we can represent any graph $G(V, E) \in I$ with vertex set $V = \{ 1, 2, \ldots, n \}$ and edge set E by a string $x_{1,2}\, x_{1,3} \cdots x_{i,j} \cdots x_{n-1,n}$ of length $n(n-1)/2$, in which $x_{i,j} = 1$ if and only if $\{ i, j \}$ is an edge of G.

In the very informal description of phase-transition phenomena given above, reference was made to varying 'parameters of instances' in order to occasion changes in behaviour. For the specific example of 3-*SAT* this parameter was the ratio between the number of clauses (m) and the number of propositional variables (n). We thus have two values associated with an instance of a decision problem, and we shall use the terminology *size* and *weight* to describe these. In 3-*SAT* the size of an instance is the number of propositional variables, the weight of an instance the number of clauses. For graph-theoretic decision problems, common choices of size and weight are the number of vertices and number of edges respectively. For a given decision problem, $f : I \rightarrow \{ 0, 1 \}$, we capture these concepts by introducing the notations below:

$$\sigma : I \rightarrow \mathbf{N} \quad \textit{The size of an instance}$$

$$\tau : I \rightarrow \mathbf{N} \quad \textit{The weight of an instance}$$

We put some restrictions on these.

a) $\exists\, \varepsilon > 0,\ k \in \mathbf{N}\ \forall\, x \in I\ |x|^{\varepsilon} \leq \sigma(x) \leq |x|^{k}$.

b) $\exists\, k \in \mathbf{N}\ \forall\, x \in I\ \tau(x) \leq |x|^{k}$.

c) $\exists\, k \in \mathbf{N}\ \forall\, n\ |\{ m : \sigma(x) = n \textit{ and } \tau(x) = m \}| \leq n^{k}$

The first two conditions state that for any instance, the size and weight functions cannot be 'too large' in terms of the length of the encoding of the instance, and, additionally, the size of an instance cannot be 'too small' in terms of the encoding length. The final condition requires that there are not 'too many' different weights that instances of a given size could have.

Apart from these, no other conditions are imposed on how the size and weight functions may be chosen for a given set of instances of a decision problem. Our main reason for, initially, allowing such freedom is that although, the measures of size and weight are present in all existing studies these have examined specific problems rather than decision problems as a class: in such cases the form of instances usually suggests 'natural' instantiations of σ and τ, cf the examples for instances of 3-*SAT* and graph-theoretic problems. In principle, however, one might have graph-theoretic problems for which phase-transitions could not be demonstrated by using the 'natural' weight function of 'number of edges' but could be exhibited with some other choice: the *NP*-complete problem of deciding if a cubic graph is 3-*edge*-colourable may well provide an exemplar of such behaviour (since all n-vertex cubic graphs have exactly $3n/2$ edges). Given the rich diversity in the forms of problems and instances, it may not be apparent what a suitable choice of weight function is. Some of the problems arising in allowing too much freedom will be illustrated in the next section.

We now argue that a statement such as 'f has a phase-transition', even when σ and τ have been defined, is insufficiently detailed. Specifically, two important aspects are missing: the measure with respect to which a 'phase-transition' is present; the precise nature of the 'dramatic shift' in this that is required. A claim that 'such-and-such a function *does not* exhibit a phase-transition' is effectively meaningless unless it is qualified with respect to a specific measure of the behaviour of a problem on instances with size n and weight m, e.g. the proportion of such instances, x, for which $f(x) = 1$; the run-time of an algorithm to decide f for such instances, etc. There are, of course, a number of ways in which this measure could be chosen and the corresponding 'dramatic shift' in behaviour defined. We introduce two general forms which will be considered subsequently.

Definition 2: Let (f, σ, τ) be as defined above, I_n denote the set of instances of f having $\sigma(x) = n$ and $I_{n,m}$ the subset of these having weight m. Define $\mu_f(n, m)$ by,

$$\mu_f(n, m) = \frac{|\{ x \in I_{n,m} : f(x) = 1 \}|}{|I_{n,m}|}$$

This corresponds to the probability that an instance x chosen uniformly at random from $I_{n,m}$ is such that $f(x) = 1$. We say that (f, σ, τ) has a *sharp phase-transition in probability*, if there exists a strictly increasing function $\theta : \mathbf{N} \rightarrow \mathbf{N}$ and a constant c such that: $\forall\, \varepsilon > 0$,

$$\lim_{n \rightarrow \infty} \mu_f(n, \lfloor (c - \varepsilon)\theta(n) \rfloor) = 0 \ (resp.\ 1)$$
$$\lim_{n \rightarrow \infty} \mu_f(n, \lceil (c + \varepsilon)\theta(n) \rceil) = 1 \ (resp.\ 0)$$

A related, but weaker, form is that of a *weak phase-transition in probability*, which is said to hold of (f, σ, τ) if, there is a strictly increasing function $\theta : \mathbf{N} \rightarrow \mathbf{N}$ such that

$$\lim_{n \rightarrow \infty} \mu_f(n, \psi(n)) = 0(1) \quad \text{if} \quad \psi(n) = o(\theta(n))$$
$$\lim_{n \rightarrow \infty} \mu_f(n, \psi(n)) = 1(0) \quad \text{if} \quad \psi(n) = \omega(\theta(n))$$

•

It may be noted that whether (f, σ, τ) has a phase-transition in probability is a purely *combinatorial* question, i.e. it is unaffected by whichever algorithmic approach is used to solve f. We can, equally, describe *algorithmic* properties. Below, this idea is developed with respect to general complexity measures and complexity classes defined on such. In order to unify the presentation we require some further notations.

Let (f, σ, τ) be given, $\theta : \mathbf{N} \rightarrow \mathbf{N}$ be a strictly increasing function, and $\alpha > 0$ be a constant. The (α, θ)-*slice* of instances, denoted $I^{(\alpha,\theta)}$, is

$$\bigcup_{n=0}^{\infty} I_{n, \lceil \alpha\theta(n) \rceil}$$

The (α, θ)-*restriction* of f, denoted $f^{(\alpha,\theta)}$ is the decision problem f restricted to instances belonging to $I^{(\alpha,\theta)}$.

Definition 3: Let Φ be a complexity measure and C_Φ an associated complexity class. We say that (f, σ, τ) has a *phase-transition* in Φ *with respect to* C_Φ if there are a strictly increasing function $\theta : \mathbf{N} \rightarrow \mathbf{N}$ and constants $c < d$ such that

$$\forall\, \alpha \in (c, d)\ f^{(\alpha,\theta)} \in C_\Phi \ \textit{and}\ \forall\, \alpha \notin [\, c, d\,]\ f^{(\alpha,\theta)} \notin C_\Phi$$
$$\textit{or}$$
$$\forall\, \alpha \in (c, d)\ f^{(\alpha,\theta)} \notin C_\Phi \ \textit{and}\ \forall\, \alpha \notin [\, c, d\,]\ f^{(\alpha,\theta)} \in C_\Phi$$

•

The definition above is given in terms of arbitrary complexity measures and classes to re-emphasise the point that, in principle, it is possible for what are regarded as phase-transition effects, to arise in many different circumstances. There are, however, a number of concrete instantiations which we shall concentrate on. The most widely studied complexity measure has been that of (worst-case) *run-time* as defined with respect to the number of steps executed by a program on some formal model of computation such as a Turing Machine. The phase-transition effects that are of interest are those which involve a dramatic reduction in the run-time of a search method when problem instances fall within (or outside) some region. Thus, Definition 3 captures both 'easy-hard-easy' and 'hard-easy-hard' patterns of behaviour by setting a given complexity class (C_Φ) as a performance 'bench-mark': the *threshold function*, θ, and associated constants (c, d) identify a region within which the decision problem f is 'much harder' or 'much easier' to solve than for instances outside it. For *NP*-complete problems, an important issue is whether choosing the complexity class to be P (deterministic polynomial time), suffices to induce a phase-transition in search. We generalise this important concept by,

Definition 4: Let $T_M : \Sigma^* \to \mathbf{N}$ denote the run-time of a program M on a given input x. $f^{(\alpha,\theta)}$ has *worst-case* time complexity $G(n)$ if there is a program M recognising $f^{(\alpha,\theta)}$ and such that:

$$\forall\ n \max_{x \in I_{n,\alpha\theta(n)}} T_M(x) \leq G(n)$$

Let $\nu : I_{n,\alpha\theta(n)} \to [\,0, 1\,]$ be a computable probability distribution with sample space $I_{n,\alpha\theta(n)}$, i.e. the set of triples

$$\{\ (x, y, z) : x \in I_{n,\alpha\theta(n)},\ y \in \mathbf{Q}^+,\ z \in \mathbf{Q}^+\ y < \nu(x) < z\ \}$$

is recursive, then $f^{(\alpha,\theta)}$ has *average-case* time complexity $G(n)$ with respect to ν if there is a program M recognising $f^{(\alpha,\theta)}$ and such that:

$$\forall\ n\ \ T^{\nu}_{av}(\,n\,) =_{def} \frac{\sum_{x \in I_{n,\alpha\theta(n)}} T_M(x)\,\nu(x)}{\sum_{x \in I_{n,\alpha\theta(n)}} \nu(x)} \leq G(n)$$

P is the complexity class (i.e set) of decision problems having worst-case time complexity within $\bigcup_{k=1}^{\infty} n^k$. For a computable distribution, ν, (AvP, ν) is the complexity class of problems having average-case time complexity n^k with respect to ν. $(\,f, \sigma, \tau\,)$ has a *phase-transition in worst-case time with respect to P* if there are $\theta : \mathbf{N} \to \mathbf{N}$ and constants $c < d$ such that

$$\forall\ \alpha \in (c, d)\ f^{(\alpha,\theta)} \in P \textit{ and } \forall\ \alpha \notin [\,c, d\,]\ f^{(\alpha,\theta)} \notin P$$
$$or$$
$$\forall\ \alpha \in (c, d)\ f^{(\alpha,\theta)} \notin P \textit{ and } \forall\ \alpha \notin [\,c, d\,]\ f^{(\alpha,\theta)} \in P$$

The definition of phase-transition in average-case time with respect to (AvP, ν) is similar. •

It is useful to develop our concept of phase-transition in worst-case time with respect to P, firstly, to take account of the fact that a proof that an NP-complete problem possessed such a property would separate P and NP; and, secondly to provide an analogue to the idea of a 'weak' phase-transition in probability. Thus, instead of the very strong condition $f^{(\alpha,\theta)} \notin P$, one might use $f^{(\alpha,\theta)}$ is NP-hard. In the context of the second extension, we say that f has a weak phase-transition in run-time with respect to P, if there are functions $\phi : \mathbf{N} \to \mathbf{N}$ and $\psi : \mathbf{N} \to \mathbf{N}$ such that $\forall\ n : \phi(n) < \psi(n)$ and

$$\forall\ m \in [\,\phi(n), \psi(n)\,]\ f^{(m)} \in P \textit{ and } \forall\ m \notin [\,\phi(n), \psi(n)\,]\ f^{(m)} \notin P$$
$$or$$
$$\forall\ m \in [\,\phi(n), \psi(n)\,]\ f^{(m)} \notin P \textit{ and } \forall\ m \notin [\,\phi(n), \psi(n)\,]\ f^{(m)} \in P$$

(where $f^{(m)}$ denotes f restricted to the subset of instances of size n having weight m).

The requirements for an algorithm to have polynomial average-case run-time with respect to, for example the uniform distribution, Λ, are technically quite strong. In fact, examining the structure of experimental studies, whereby random instances of specific sizes and weights are generated, motivates a slightly weaker concept of 'run-time of a typical instance'. This concept, which we call, 'probable run-time' is described by,

Definition 5: Let Q be a program recognising $f^{(\alpha,\theta)}$ and

$$Bad_Q(\,n, G(n)\,) = \{\ x \in I_{n,\alpha\theta(n)} : T_Q(x) > G(n)\ \}$$

$f^{(\alpha,\theta)}$ has *probable run-time* $G(n)$ if there is a program M recognising $f^{(\alpha,\theta)}$ such that

$$\lim_{n \to \infty} \frac{|Bad_M(\, n, G(n)\,)|}{|I_{n,\alpha\theta(n)}|} = 0$$

pP is the complexity class of problems having probable run-time n^k. •

We note that while it can be established that $P \subseteq (AvP, \Lambda) \subseteq pP$, it may not be the case that $f \in pP$ implies $f \in (AvP, \Lambda)$: consider a problem, f, in which there are 2^n instances of size n and for which one has an algorithm that takes $2^{3n/4}$ steps on $2^{n/2}$ instances and n steps on the remaining $2^n - 2^{n/2}$; it is clearly the case that $f \in pP$, but the algorithm used to demonstrate this has average-case run-time with respect to Λ exceeding $2^{n/4}$.

In summary, formal models of phase-transition behaviour for classes of decision problem must involve a number of elements: defined mechanisms for parameterising instances of a problem (the size and weight functions: σ and τ); a precise sense of the measure in which a phase-transition effect is being considered and a concept of what changes in this measure constitute significant effects.

3. Combinatorial and Algorithmic Properties

Two things are apparent from the formal structures described in the previous section and particularly evident in the definitions of 'phase-transition' proposed: each of the components articulated is present in earlier studies of particular problems; secondly, the degree of licence that is allowed in instantiating these structures for a given combinatorial problem, indicates that statements such as 'f does *not* have a phase-transition' are likely to be impossible to establish for a given measure, *except* when specific size and weight functions are provided.

In this section we consider the following question, with particular reference to f being NP-complete.

> What, if anything, can be deduced about phase-transitions in run-time from the property of $(\, f, \sigma, \tau\,)$ having a phase-transition in probability?

We observed when introducing the concept of phase-transition in probability that its definition was independent of algorithmic considerations. The following results emphasise that a purely combinatorial property suffices for such a phase-transition to exist. A decision problem, f, over $\{\,0, 1\,\}^*$ is *monotone (increasing)* if whenever $\alpha \in \{\,0, 1\,\}^n$ is such that $f(\,\alpha\,) = 1$, any $\beta \in \{\,0, 1\,\}^n$ obtained by changing a 0 to a 1 in α, also has $f(\,\beta\,) = 1$. A decision problem is monotone *decreasing* if $f(\,\alpha\,) = 0$ implies $f(\,\beta\,) = 0$ where β is obtained in the same way.

Fact 1: A decision problem $f : \{\,0, 1\,\}^* \to \{\,0, 1\,\}$ is said to be *trivial* if either $\forall\, x\; f(x) = 0$ or $\forall\, x\; f(x) = 1$. A decision problem, f, on graphs is *symmetric* if $f(G(V, E)) = 1$ if and only if $f(H(V, F)) = 1$ for any H that is isomorphic with G.

a) If $f : \{\,0, 1\,\}^* \to \{\,0, 1\,\}$ is a non-trivial monotone decision problem, then $(\, f, |x|, H(x)\,)$ has a weak phase-transition in probability.

b) If f is a symmetric monotone decision problem defined on graphs, $G(V, E)$ then $(\, f, |V|, |E|\,)$ has a sharp phase-transition in probability.

Proof: a) is from (Bollobás and Thomason, 1987); b) from (Friedgut and Kalai, 1996). □

That 'monotonicity' is not a *necessary* condition, even when restricting size and weight functions to be $|V|$ and $|E|$ is easily demonstrated by the result below.

Theorem 1: For $k \geq 2$, $ISO-HAM_k$ is the decision problem defined on n-vertex graphs $G(V, E)$ as follows: $ISO-HAM_k(\,G\,) = 1$ if and only if there is a subset W of the vertices V such that $|W| \geq n - k$; the sub-graph $G[W]$ of G induced by W contains a Hamiltonian cycle; and, the vertices $V - W$ form an independent set in $G(V, E)$. For all fixed $k \geq 2$, the decision problem $ISO-HAM_k$ is non-monotone and $(ISO-HAM_k, |V|, |E|)$ has a sharp phase-transition in probability with $\theta(n) = n \log_e n$ and $c = 0.5$.

Proof: To show that $ISO - HAM_k$ is not monotone it suffices to define a sequence of three n-vertex graphs G_0, G_1, G_2 such that G_i is a sub-graph of G_{i+1} $(0 \le i < 2)$ and for which $ISO - HAM_k(G_0) = ISO - HAM_k(G_2) = 0$ but $ISO - HAM_k(G_1) = 1$. Define G_0 to be a graph consisting of a path of $n - k$ vertices and k isolated vertices. Obviously $ISO - HAM_k(G_0) = 0$. Form G_1 by adding an edge, to G_0, joining the two vertices at the ends of the $n - k$ vertex path. Now $ISO - HAM_k(G_1) = 1$, since the additional edge completes a Hamiltonian cycle with $n - k$ vertices. Finally, form G_2 from G_1 by adding an edge between any two of the k isolated vertices in G_1. We now have $ISO - HAM_k(G_2) = 0$ since the vertices of this graph cannot be partitioned into an independent set of size k *and* a Hamiltonian inducing set of size at least $n - k$. For the second part of the theorem, first note that for n large enough, any n-vertex graph, $G(V, E)$ with a Hamiltonian cycle satisfies $ISO - HAM_k(G) = 1$: simply choose $W = V$. In addition, any n-vertex $G(V, E)$ with at least $k + 1$ isolated vertices must have $ISO - HAM_k(G) = 0$. The analysis of Koršunov (1976) show that for all $\varepsilon > 0$, almost all graphs with $(0.5 + \varepsilon)n \log_e n$ edges are Hamiltonian. On the other hand, for any fixed k, the result of Erdös and Rényi (1959) indicates that almost all n-vertex graphs with $(0.5 - \varepsilon)n \log_e n$ edges have $k + 1$ isolated vertices. □

We now present an explicit definition of an NP-complete problem, f, such that (f, σ, τ) has a phase-transition in probability, but (f, σ, τ) does *not* have a phase-transition in worst-case time with respect to P. This construction, therefore, gives a partial answer to the question raised at the start of this section..

In fact, we prove this result for 3-SAT. For a 3-CNF formula, F over a set of n propositional variables, we take the size of F to be the number of variables, i.e. $\sigma(F) = n$, and the weight of F to be the number of clauses, m, defining F.

Theorem 2: a) $(3 - SAT, n, m)$ has a sharp phase-transition in probability. b) $(3 - SAT, n, m)$ does not have a phase-transition in worst-case time with respect to P.

Proof: a) Immediate from the results of (Friedgut and Kalai, 1996).

b) We prove this for any strictly monotone increasing function, $\theta : \mathbf{N} \to \mathbf{N}$ and any constant c satisfying:

$$\forall n \lceil c\theta(n) \rceil \ge n/3 \quad and \quad \lim_{n \to \infty} \frac{c\theta(n)}{7\binom{n}{3}} < 1 \tag{1}$$

by showing that the problem $3 - SAT^{c,\theta}$ is NP-complete. Recall that this decision problem is defined as:

Instance: A formula, F in 3-*CNF* defined over a set of n propositional variables *and* containing *exactly* $\lceil c\,\theta(n) \rceil$ distinct clauses.

Question: Does there exist an instantiation of the n variables that satisfies F?

We note that if $m < n/3$, then a 3-CNF F with m clauses cannot essentially depend on n variables (each clause accounts for exactly three distinct variables), hence the case $\theta(n) < n/3$ is redundant. In addition if $m > 7\binom{n}{3}$ then F cannot be satisfiable: any instantiation α of n variables fails to satisfy exactly $\binom{n}{3}$ clauses of size 3, any so for any instantiation α, F must contain at least one clause that is not satisfied by α.

We prove that *3-SAT* is polynomially reducible to $3 - SAT^{c,\theta}$ for all θ, c satisfying (1). Let

$$F = C_1 \wedge C_2 \cdots \wedge C_m$$

be an instance of 3-*SAT* over $X_n = < x_1, \ldots, x_n >$ (where, without loss of generality, it may be assumed that the clauses of F are all distinct). We construct (in time polynomial in n) a 3-*CNF*, $\rho(F) = H$ with the following properties:

a) H is defined over $4n^3$ propositional variables.

b) H contains exactly $\lceil c\,\theta(4n^3)\rceil$ distinct clauses.

c) H is satisfiable if and only if F is satisfiable.

Let $N = 4n^3 - n$. The propositional variables of H are $X_n \cup Y_N$ where

$$Y_N = < y_1, y_2, \ldots, y_N >$$

The clauses of H are formed from the m clauses of F and a set of clauses chosen by taking all possible clauses of 3 literals over Y_N *except for*

d) clauses with all three variables negated.

e) t additional clauses.

Here t is a value that is set subsequently and depends only on m, θ and c. Thus, the number of clauses in H is exactly

$$m + 7\binom{N}{3} - t.$$

It is easy to see that H is satisfiable if and only if F is satisfiable: a satisfying assignment to H must satisfy F since the clauses of F are a subset of those of H; on the other hand if F is satisfiable, then extending the assignment to X_n that satisfies F by the assignment $< y_i := 1 >$ for each $1 \le i \le N$ gives a satisfying assignment for H. All that remains is to show that t can be chosen from the range $0 \le t \le 7\binom{N}{3}$ so that:

$$t = m + 7\binom{N}{3} - \lceil c\,\theta(n+N)\rceil \qquad (2)$$

The expression on the right-hand side of (2) is at most,

$$8\binom{n}{3} + 7\binom{N}{3} - \frac{n+N}{3}$$

which is certainly less than the maximum value allowed for t since $n + N = 4n^3$. Similarly, the right-hand side of (2) is at least

$$7\binom{N}{3} - \lceil c\,\theta(n+N)\rceil$$

and the condition on the maximum value of $c\,\theta(n+N)$ suffices to ensure that for n greater than some constant value (depending only on c and θ), that this expression is greater than 0. It is clear that the construction of H from F can be accomplished in time polynomial in $n + m$. □

The 'padding' technique that is used to establish Theorem 2, is quite general, and can be applied to 3-*COL*, variants of *k-clique*, etc.

One immediate conclusion that can be drawn from Theorem 2 is that the apparent speed-up in search time occurring for some regions in empirical studies of random *3-SAT* instances, is *not* caused by a specific search method taking polynomial-time in these regions. There is, of course, the possibility that this behaviour indicates probable polynomial time (in the sense of Definition 5) or even average-case polynomial time. If it is the former, then the question arises as to how many 'bad' instances can occur in the 'easy' regions? It has been observed, experimentally, with respect to specific search methods for *3-SAT* that on occasion random instances in the 'easy' region can take considerable time to solve, e.g. (Freeman, 1996). Our next result shows that (unless $P = NP$) any procedure for *3-SAT*$^{(c,n)}$ that has polynomial *probable* run-time must be such that for any constant r and k there are infinitely many n for which more that n^r instances in each $(n, \lceil cn \rceil)$-slice require run-time exceeding n^k.

Theorem 3: Suppose that Q is an algorithm for $3 - SAT^{(c,n)}$ $(c \geq 1)$ such that Q has polynomial probable run-time n^k. $P = NP$ if Q satisfies $\exists\, r\; \forall\, n \;\; |Bad_Q(n, n^k)| \leq n^r$.

Proof: Suppose the contrary and that $P \neq NP$ but that Q is such that

a) Q has probable run-time n^k.

b) $\forall\, n$ there are at most n^r n-variable, $\lceil cn \rceil$-clause instances on which Q takes more than n^k steps.

We describe an algorithm, S, that decides any n-variable, m-clause instance, F, of 3-*SAT* in polynomial time using Q. This would only be possible if $P = NP$.

Input: $F(X_n)$, n-variable, m-clause instance of 3-*SAT*.
Output: 1 if F is satisfiable; 0 otherwise.
$count := 0$;
while $count \leq (4n^3)^r + 1$ **loop**
  $count := count + 1$; $H_{count} := next(\rho(F), count)$;
  Run Q on H_{count} for *at most* $(4n^3)^k + 1$ steps;
  if Q has finished **then**
    return the result given by Q;
    stop;
  end if;
end loop;

In this algorithm $next(\rho(F), count)$ returns a $(4n^3)$-variable, $\lceil c(4n^3) \rceil$-clause 3-*CNF* that is constructed using the method in the proof of Theorem 2. The only change is that on each successive construction (as recorded by the value in *count*) a *different* set of t clauses is removed from the available clauses (over the Y_N variables) so that the instances H_i and H_j are the same if and only if $i = j$. From the construction used in Theorem 2, the required value of t is

$$7\binom{N}{3} + m - \lceil c(n+N) \rceil$$

It follows that the total number of *different* instances that could be formed in this way is $\exp(\Omega(n))$. Since, we have assumed that there are at most $(4n^3)^r$ 'bad' instances for Q it follows that after at most $(4n^3)^r + 1$ iterations of the main loop in the algorithm an instance will have been constructed on which Q has finished after $(4n^3)^k$ steps. The correctness of the answer returned follows from Theorem 2, and the total running time of S is clearly polynomially bounded. It follows that if an algorithm with the properties claimed of Q existed then we could construct a (worst-case) polynomial time algorithm for 3-*SAT*. This contradicts the initial premise that $P \neq NP$. □

Although Theorem 3 does not imply that *all* values of n have many 'bad' instances, it is easy to see (assuming $P \neq NP$) that it implies the existence of infinitely many such n. Suppose, however, for a fixed r and algorithm Q that $[\, b_i \,]$ is the (infinite) sequence of values such that $|\, Bad_Q(b_i, (b_i)^k)| > (b_i)^r$: unless $P = NP$, Theorem 3 guarantees the existence of such a sequence for any Q and any $r \in \mathbf{N}$. In principle, it could be the case that b_{i+1} is 'much' greater than b_i, e.g. suppose that $b_1 = 3$ and for $i > 1$ $b_i = 2^{n_{i-1}}$, then, with such an algorithm one would be able to solve instances of 3-*SAT* quickly for all instance sizes, n, $4n^3 \neq b_i$ for some i. This might well be considered a feasible algorithm, since its performance would be fast for 'long' sequences of values of n for which $| Bad_Q(n, n^k) |$ was at most n^r. Such an algorithm would *not* contradict the premise $P \neq NP$ since its run-time would not be polynomial time for arbitrarily large values of n.

It turns out, however, that this is not possible. For a given Q and r as above, let $[g_i]$ be the sequence of instances sizes such that $|Bad_Q(g_i, (g_i)^k)| \leq (g_i)^r$. The following development of Theorem 3 establishes for any such sequence there must be infinitely many points i such that $g_i - g_{i-1}$ is 'large'. More formally, let $s = [n_i]$ denote a strictly monotone increasing sequence of natural numbers,

$$s \ =_{def} \ < n_0, n_1, \ldots, n_i \cdots >$$

We say that such a sequence is *p-good* if there exists $p \in \mathbf{N}$ such that for all $i > 0$ it holds that $n_i < (n_{i-1})^p$.

Theorem 4: Let Q be an algorithm for $3 - SAT^{(c,n)}$ with polynomial probable run-time n^k. $P = NP$ if there exists a p-good sequence $[n_i]$ and a constant $r \in \mathbf{N}$ such that

$$\forall \ i \ \ | Bad_Q(n_i, (n_i)^k) | \ \leq \ (n_i)^r$$

Proof: The proof is again by contradiction. Assume that $P \neq NP$ and that Q is an algorithm for $3 - SAT^{(c,n)}$ with probable polynomial run-time n^k. Further assume that there is a p-good sequence $[n_i]$ and a constant $r \in \mathbf{N}$ for which

$$\forall \ i \ \ | Bad_Q(n_i, (n_i)^k) | \ \leq \ (n_i)^r$$

The algorithm, S, below evaluates any n-variable instance of 3-SAT in polynomial time. It should be noted that the polynomial bound and the correctness of the algorithm do not require the sequence $[n_i]$ *to be known*: its existence suffices for the method below to run in time bounded by some polynomial function of n.

```
Input: F an instance of 3-SAT over n variables.
Method:
  m := 4n^3; s := m^r + 1;
  Pool := { H_1 ; H_2 ; ··· ; H_s };
  -- each H_i is a member of R(F), i.e. an instance of 3 − SAT^(c,m) as constructed
  -- by the reduction of Theorem 2.
  i := 0;
  loop
    for each H ∈ Pool loop
      Run Q on H for m^k + 1 steps;
      if Q( H ) has halted then
        return Q(H) ; stop;
      end if;
    end loop;
    -- if this point is reached then m ∉ [ n_i ]
    Old_Pool := Pool ; Pool := { };
    i := i + 1 ; m := m + 1 ; s := m^r + 1 ; j := 1;
    --************************************
    while | Pool | < s loop
      Pool := Pool ∪ { H_j^1 , . . . , H_j^(m−1) };
      Old_Pool := Old_Pool − { H_j } ; j := j + 1;
    end loop;
    --************************************
  end loop;
 end;;
```

The procedure first of all constructs $m^r + 1$ different instances of $3 - SAT^{(c,m)}$, where $m = 4n^3$, using the reduction of Theorem 2. If each belongs to $Bad_Q(m, m^k)$ then the **while** loop that follows replaces each instance with m different instances of $3 - SAT^{(c,m+1)}$ - the H_j^t of the loop body. Let $< x_1 , \ldots , x_m >$ be the variables defining H_j then each H_j^t is formed by choosing at most $\lceil c \rceil$ clauses of the form $z \vee x_a \vee x_b$, where z is a new variable. Since there are $m(m-1)/2$ such clauses to choose from, it is certainly the case that $m - 1$ different subsets of size at most $\lceil c \rceil$ can be selected. Furthermore, the resulting H_j^t are all distinct and each is satisfiable if and only if H_j is satisfiable, i.e. if and only if F is satisfiable.

This establishes that the algorithm correctly evaluates 3-$SAT(F)$, leaving only its polynomial time complexity to be demonstrated. Consider the p-good sequence $[n_i]$ assumed to exist for Q. Each iteration of the outer loop deals with $m^r + 1$ different instances of $3 - SAT^{(c,m)}$ where m is initially $4n^3$ and increases by one at each iteration. Clearly there must be some point, i, of the p-good sequence such that $n_i < 4n^3 \leq n_{i+1} < (n_i)^p$. It follows that the total number of iterations of the outer loop is at most $(4n^3)^p$. In the main body requires at most $(m^k + 1)$ steps are performed on at most $m^r + 1$ instances. >From the previous argument $m \leq (4n^3)^p$. Thus, the run-time of the entire procedure is $O(n^{3p(k+r+1)})$ where p, k and r are constants. This contradicts our initial premise that $P \neq NP$, hence the Theorem follows. □.

Theorem 3 relies on there being sufficiently many different ways of ‘shrinking’ the initial padded instance of F in order to obtain an instance with exactly the right number of clauses (the restriction to $\theta(n) = n$ is not necessary: this is done solely to simplify the proof). Similar arguments, however, can be made for the other NP-complete problems to which the techniques of Theorem 2 can be adapted.

In fact, we can show that if $|Bad_Q(n, n^k)|$ is sub-exponential in n, then it is possible to construct a *randomised* algorithm for 3-SAT which for *any* instance has polynomial run-time with overwhelmingly high probability. For a given instance, F of 3-SAT on n variables, let $R(F)$ denote the set of all instances of $3-SAT^{(c,N+n)}$ onto which F could be mapped using the reduction of Theorem 2, where $N = 4n^3 - n$.

Theorem 5: Let Q be an algorithm for $3-SAT^{(c,n)}$ with polynomial probable run-time n^k. If there is a constant $r \in \mathbf{N}$, with which for all n and all n-variable instances, F, of 3-SAT it holds that

$$|\, Bad_Q(4n^3, n^k)\,| \leq |R(F)|\,(1 - n^{-r})$$

then there is a (randomised) algorithm S for 3-SAT such that for all F

$$P\,[\,T_S(F) > n^{r+1}(n^k + 1)\,] < \exp(-n)$$

Proof: The method is similar to that in Theorem 3, however: the upper limit of the **while** loop is changed to n^{r+1}; H_{count} is chosen uniformly at random from the set $R(F)$, each successive choice being made independently of previous choices; and, 3-$SAT(F)$ is evaluated explicitly if the **while** loop exits. In order for the run-time on an instance F of size n to exceed $n^{r+1}(n^k + 1)$ every random choice of H_{count} must belong to $Bad_Q(4n^3, n^k)$. The probability of this is at most

$$\left(\frac{|Bad_Q(4n^3, n^k)|}{|R(F)|} \right)^{n^{r+1}} < (1 - n^{-r})^{n^{r+1}}$$

Since, $1 - x < \exp(-x)$, it follows that this probability is at most $\exp(-n)$. □

To conclude this section, we illustrate one particular rather weak algorithmic 'benefit' that results directly from the combinatorial *proofs* in a number of phase-transition results. Consider the decision problem *Sequential Truth Assignment* (*STA*)

Instance: $F(X_n, Y_n)$ a 3-CNF formula defined over a set of $2n$ Boolean variables.

Question: $\exists\,\alpha_1\,\forall\,\beta_1\,\exists\,\alpha_2\,\forall\beta_2 \cdots \exists\alpha_n\,\forall\beta_n\; F(\alpha_1, \alpha_2, \ldots, \alpha_n\,;\beta_1, \beta_2, \ldots, \beta_n) = 1$?

STA can be interpreted as a 2-person game involving players X and Y who at the k'th round of the game choose a Boolean value for the variables x_k and y_k. X wins if the value of F after all choices have been completed is 1; otherwise Y wins. The decision problem *STA* asks if X has a forced win with the given instance, i.e. 3-CNF, of the game. Stockmeyer and Meyer (1973) have shown that *STA* is *PSPACE*-complete.

Theorem 6: Let $\theta : \mathbf{N} \to \mathbf{N}$ be any monotone increasing function such that $\theta(n) \underset{n}{\to} \infty$. For a $2n$ variable, m clause instance F of *STA*, define $\sigma(F) = 2n$ and $\tau(F) = m$.

$$\lim_{n \to \infty}\; \mu_{STA}(2n, \theta(n)) \;=\; 0.$$

Proof: If F is an instance of *STA* that contains a clause defined by three distinct variables in Y_n, then $STA(F) = 0$: player Y can force F to take the value 0 just by setting these variables appropriately. Call such a clause a Y-clause of which there are $8\binom{n}{3}$. To prove the theorem it suffices to estimate the number of clauses, m, needed to ensure that an instance chosen uniformly at random from the set of $2n$-variable, m-clause instances almost certainly contains a Y-clause. Let $F_{n,m}$ be such a random instance.

$$P\,[\,STA(\,F_{n,m}\,) = 0\,] \;\geq\; P\,[\,F_{n,m}\ has\ a\ Y - clause\,]$$

$$= 1 - P\,[\,F_{n,m}\ does\ not\ have\ a\ Y - clause\,]$$

$$\geq 1 - \exp\left\{ - \frac{mn(n-1)(n-2)}{2n(2n-1)(2n-2)} \right\} \approx 1 - \exp(-m/8\,).$$

Thus if $m = \theta(n) \underset{n}{\rightarrow} \infty$, then almost certainly $F_{n,m}$ contains a Y-clause and $STA(\,F_{n,m}\,) = 0$. □

Corollary 1: STA $\in Av(\,P, \Lambda\,)$ (where Λ is the uniform distribution).

Proof: Consider the algorithm, S, which evaluates an instance, $F(\,X_n, Y_n\,)$ of STA by checking if F has a Y-clause. If a Y-clause is found then the result 0 is returned, otherwise an exhaustive search is used to determine the answer. The exhaustive search takes at most $2^{2n}\,m$ steps to classify a $2n$-variable, m-clause instance. Let $|I_{2n}|$ denote the number of possible instances that can be formed from $2n$ variables X_n and Y_n; $|F|$ denote the number of clauses in F; N the number of possible clauses; and, $\gamma : \mathbf{N} \rightarrow \mathbf{N}$ be a function which will be fixed subsequently. The average run-time $T(n)$ of S is,

$$T(\,n\,) = \sum_{F \in I_{2n}} T_S(\,F\,)/|I_{2n}|$$

$$\leq \sum_{F : |F| \geq \gamma(n)\ and\ F\ has\ a\ Y-clause} |F|/|I_{2n}| + \sum_{F : |F| \geq \gamma(n)\ and\ F\ has\ no\ Y-clause} 2^{2n}\,|F|/|I_{2n}|$$

$$+ \sum_{F : |F| < \gamma(n)} 2^{2n}|F|/|I_{2n}|$$

From the proof of Theorem 6, a fraction of at least $1 - \exp(-\gamma(n)/8\,)$ of the instances with $\gamma(n)$ or more clauses, has a Y-clause. Hence,

$$T(\,n\,) \leq \sum_{m=\gamma(n)}^{N} m + \exp(-\gamma(n)/8\,)\,2^{2n}\,N + P[\,|F| < \gamma(n)]\,2^{2n}\,\gamma(n)$$

$$\leq N^2 + \exp(\,2(\log_e 2)n + \log_e N - \gamma(n)/8\,) + P[|F| < \gamma(n)]\,2^{2n}\,\gamma(n)$$

If $\gamma(n) \geq 16(\log_e 2)n$ then the $\exp(\cdots)$ term is at most N. In addition, a simple calculation shows that $P\,[|F| < 16(\log_e 2)n]$ is less than 2^{-2n}. Thus, $T(\,n\,)$ is

$$\leq N^2 + N + 16(\log_e 2)n = O(\,n^6\,)$$ □

We have made no attempt to reduce the degree of the polynomial bound given, for the simple reason that the algorithm described is of little interest as a mechanism for deciding Sequential Truth Assignment: it is merely testing if a simple condition sufficient to return the result 0 holds.

4. Size and Weight Functions

We now turn to the problems of distinguishing ‘reasonable’ weight functions. A simple example illustrates why some care is necessary. Consider the *parity* function on finite binary strings, defined as $PAR(\,x\,) = 1$ if and only if the number of bits equal to 1 in x is even. Suppose that we choose the size function for instances of parity to be the length of the string: $\sigma(\,x\,) = |x|$. If the ‘most natural’ choice of weight function is used, i.e. $\tau(\,x\,) = H(x)$, where $H(x)$ denotes the number of 1s in x, then it is clear that $(PAR, |x|, H(x))$ does not exhibit a phase-transition in probability. Moreover, since $PAR(x)$ can be evaluated in exactly $|x|$ steps, it is not possible to induce a phase-transition in run-time. Therefore, one might conclude ‘parity does not have a phase-transition (in probability)'. This, however, within the terms permissible both by the formulation above and by previous studies (since not explicitly prohibited), is not the case.

Lemma 1: There is a choice of size and weight functions, σ and τ, such that (PAR, σ, τ) has a phase-transition in probability using $\theta(n) = n + 1$ and $c = 0.5$.

Proof: Choose $\sigma(x) = |x|$ as before, and $\tau(x)$ to be

$$1 + \lceil |x|/2 \rceil + \lceil H(x)/2 \rceil (1 - 2 \{ \lceil H(x)/2 \rceil - \lfloor H(x)/2 \rfloor \})$$

If $H(x)$ is even, i.e. $PAR(x) = 1$, then $1 + \lceil n/2 \rceil \leq \tau(x) \leq n + 1$. On the other hand, if $H(x)$ is odd, then $1 \leq \tau(x) \leq \lceil n/2 \rceil$. Hence, if we choose a random x with $|x| = n$ and $\tau(x) = m$, whenever $m/(n+1) < 0.5$ we have $PAR(x) = 0$; whenever $m/(n+1) > 0.5$ we have $PAR(x) = 1$. □

The parity function is in P and it may seem that the device used to induce a phase-transition in probability is merely a technical curiosity. We can, however, create a similar difficulty for NP-complete problems (albeit somewhat artificial ones). We introduce two decision problems derived from 3-*COL* (although the method could be applied to any NP-complete problem).

Definition 6: OE-3COL is the decision problem whose instances are n-vertex undirected graphs $G(V, E)$ such that $|E| \leq n(n-7)/2$. If $G(V, E)$ contains an *even* number of edges then 1 is returned if G is 3-vertex colourable; if G contains an *odd* number of edges then 1 is returned if the complement of G (i.e. the graph in which an edge $\{ i, j \}$ is present if and only if $\{ i, j \}$ is not an edge in G) is 3-vertex colourable.

Let $N(n, k)$ denote the number of n-vertex graphs with exactly k edges. Let ord_k^N ($N = n(n-1)/2$) be a bijection that assigns a unique integer, r, $1 \leq r \leq N(n, k)$ to each such graph. *Ord-3COL* is the decision problem whose instances are n-vertex undirected graphs $G(V, E)$. If $ord_{|E|}^N(G) \leq 0.5N(n, |E|)$ then 1 is returned if G is 3-vertex colourable; if $ord_{|E|}^N(G) > 0.5N(n, |E|)$ then 1 is returned if the complement of G is 3-vertex colourable. •

Theorem 7: a) $OE-3COL$ is NP-complete; b) there is an ordering bijection such that $Ord-3COL$ is NP-complete.

Proof: a) Trivial. *b)* We first construct a bijection between n-vertex, k-edge graphs and values in the range 1 to $N(n, k)$ which will allow $Ord-3COL \in NP$ to be proved. Consider the following ordering of m-bit binary strings containing exactly k 1s: if x is such a string then $ord_k^m(x)$ is the unique number lying between 1 and $\binom{m}{k}$ defined by the following rules:

$$ord_k^m(0^{m-k} 1^k) = 1 \; ; \; ord_k^m(1^k 0^{m-k}) = \binom{m}{k}$$
$$ord_k^m(0x) = ord_k^{m-1}(x) \; ; \; ord_k^m(1x) = \binom{m-1}{k} + ord_{k-1}^{m-1}(x)$$

Obviously, for any x with exactly k 1s, the value $ord_k^{|x|}(x)$ is computable in time polynomial in $|x|$. Represent an n-vertex undirected graph, $G(V, E)$ by a sequence of $N = n(n-1)/2$ bits using the methods described after Definition 1, above. Let $G(x)$ denote this encoding so that the number of edges in G is the number of bits with value 1 in $G(x)$. The ordering function required is $ord_{|E|}^N(G(x))$ where $|E|$ is the number of bits equal to 1 in $G(x)$. Since ord_k^m is polynomial-time computable and 3-*COL* is in NP, it is immediate that *Ord-3COL* is in NP. To prove it is NP-hard we show that 3-*COL* is polynomially reducible to *Ord-3COL*. Let $G(V, E)$ be an instance of $3-COL$, i.e. an n-vertex graph, $G(y)$ denote the binary encoding of the edges of G as described earlier, and N denote $|G(y)|$, i.e. $n(n-1)/2$.

If $ord_{|E|}^N(G(y)) \leq 0.5\,N(n, |E|)$ then $G(V, E)$ is 3-colourable if and only *Ord-3COL(G)* holds. Otherwise we construct a $2n$-vertex graph $H(W, F)$ with binary encoding $G(z)$ where

$|G(z)| = n(2n-1)$ as follows: the vertex w_{n+i} of W is the vertex v_i of V, where $1 \le i \le n$. The vertices $w_1, \ldots, w_n$ are new vertices. The edges are $\{ w_{n+i}, w_{n+j} \}$ where $\{ v_i, v_j \}$ is an edge of G. Thus H may be seen as relabeling the vertices of G with labels $n+1, n+2, \ldots, 2n-1, 2n$, leaving the edges unchanged, and introducing n new vertices each of which has degree 0. It is obvious that $H(W, F)$ is 3-colourable if and only if $G(V, E)$ is 3-colourable. Let M denote $n(2n-1)$, i.e. $|G(z)|$, and consider the value of $ord^M_{|F|}(G(z))$. From our construction,

$$G(z) = 0^{N+n^2}\, G(y)$$

since the degree of each of the vertices w_i $(1 \le i \le n)$ is 0. Hence, $|F| = |E|$. By definition,

$$ord^M_{|F|}(G(z)) \ = \ ord^N_{|E|}(G(y)) \ \le \ \binom{N}{|E|}$$

But, recalling that $N = n(n-1)/2$ and $M = n(2n-1)$,

$$\binom{N}{|E|} \ \le \ 0.5\binom{M}{|F|}$$

and hence $G(V, E)$ is 3-colourable if and only if $Ord - 3COL(H(W, F))$. This completes the proof. □

Theorem 8: a) $(OE - 3COL, |V|, |E|)$ does not have a phase-transition in probability. b) There is a polynomial-time computable weight function, τ, for which $(OE - 3COL, |V|, \tau(G))$ *does* have a phase-transition in probability. c) For *any* weight function, τ, that considers only $|V|$ and $|E|$, $(Ord - 3COL, |V|, \tau(G))$ *does not* have a phase-transition in probability.

Proof: (Outline) For (a), there is a constant $c < 3$ such that graphs with $|E|/n < c$ are almost certainly 3-colourable; graphs with $|E|/n > c$ are almost certainly not 3-colourable. Since the complement of a k-edge graph contains $n(n-1)/2 - k$ edges, as k increases the probability of a random k-edge graph returning 1 oscillates from ‘close to’ 1 to ‘close to’ 0 (when $k \le cn$). For (b), the method is similar to that employed in Lemma 1: use an ordering that assigns lowest weights to graphs with an even number of edges (with smaller even numbers preceding larger values) and highest weights to graphs with an odd number of edges, but with larger odd numbers having a *lower* weight than smaller odd numbers. Since the number of edges in the complement graph always exceeds $3n$, the transition from ‘almost certainly’ 3-colourable to ‘almost certainly’ not 3-colourable occurs at some point in the first half of this ordering and no further transition occurs. To prove (c) we note that the probability of a random k-edge graph having value 1 tends to 0.5 for $k \le cn$ or $k \ge n(n-1)/2 - cn$. □

5. Some Open Problems

We conclude by highlighting some open problems, starting with the relationship between existence of a phase-transition in probability and complexity-theoretic issues. It is well-known that, under the uniform distribution, some NP-complete problems can be solved by algorithms running in average-case polynomial time. Such problems include 3-SAT, e.g. (Koutsapias and Papadimitriou, 1992) and 3-COL, e.g. Wilf (1985). It is easy to show that for such methods the set of instances requiring run-time exceeding any given polynomial bound are a sparse set. Both of these examples, however, deal with problems that have a phase-transition in probability.

Question 1: Does there exist any (recursive) language $L \subseteq \{0, 1\}^*$ that has *both* of the following properties:

a) L has a phase-transition in probability.

b) L is NP-hard and (under the standard assumptions) for any algorithm Q recognising L

$\forall\, k \in \mathbf{N} \quad Bad(\, Q, k\,) =_{def} \{\, x : T_Q(\, x\,) > |x|^k \,\}$

satisfies $\lim\limits_{n\to\infty} |Bad(Q,k) \cap \{\,0,1\,\}^n|/2^n > 0$, i.e. is not sparse?

We observe that any such L, cannot belong to AvP with respect to the uniform distribution or have probable polynomial-time complexity.

'Natural' NP-complete graph problems without phase-transitions in probability

Consider any graph property, Π, that meets the following criteria:

a) infinitely many graphs satisfy Π;

b) if $G(V,E) \in \Pi$ then any graph, H that is isomorphic to G belongs to Π;

c) there are functions $\alpha : \mathbf{N} \to \mathbf{N}$ and $\beta : \mathbf{N} \to \mathbf{N}$ for which $\beta(n) - \alpha(n) \to \infty$ and

$$(\exists\, G \in \Pi : |V(G)| = n\,) \Rightarrow (\forall\, \alpha(n) \le m \le \beta(n)\ \begin{cases} \exists\, G(V,E) \in \Pi\ s.t.\ |E| = m \\ and \\ \exists\, G(V,E) \notin \Pi\ s.t.\ |E| = m \end{cases}$$

Question 2: Does there exist any (or recursive or belonging to a specified complexity class) 'natural' graph property, Π, that satisfies (a-c) above, and is such that if $\sigma(\,G\,)$ is chosen as $|V(G)|$ then there is no choice of weight function τ defined solely in terms of $|V|$ and $|E|$ for which it can be shown that (Π, σ, τ) has a phase-transition in probability?

The property $Ord-3COL$ introduced earlier fails to satisfy the isomorphism criterion; similarly properties which are defined in terms of k-regular graph structures, e.g. k-edge colourability of k-regular graphs, k-vertex colourability of $(2k-2)$-regular graphs, etc, do not satisfy (c). Notice that even employing rather artificial devices such as the following

$$G(V,E) \in \Pi \quad iff \quad |V(G)|\ is\ even\ \ and\ \begin{cases} G\ is\ k-edge\ colourable & \text{if } G\ is\ k-regular \\ G \in Q \quad (for\ some\ property\ Q) & otherwise \end{cases}$$

does not necessarily help: Q cannot be such that for any value of k either almost all graphs with $kn/2$ edges belong to Q or almost all graphs with $kn/2$ edges do not belong to Q.

Question 3: The results regarding the $PSPACE$-complete problem Sequential Truth Assignment derived in Theorem 6 and its Corollary, rely on the fact that a Y-clause is almost sure to appear in a randomly chosen formula. If these are excluded from the set available, then STA is still a monotone (decreasing) property. Suppose we denote this restricted version by $mod-STA$.

a) Does $(mod-STA, 2n, m)$ have a sharp phase-transition in probability?

b) Is $mod-STA \in Av(\,P, \Lambda\,)$?

It may be possible to address both of these by finding a simple condition which defines a large enough set of formulae, F, for which $STA(\,F\,) = 1$.

As regards these questions we discuss some candidate decision problems:

Problem 1: The natural cases to consider are the average-case NP-complete problems, i.e. those which do not have a polynomial-time average-case algorithm unless every NP problem has one under the uniform distribution. The Tiling problem of Levin (1986) was the first such problem to be identified. The problem defined by Levin differs slightly from the NP-complete Tiling problem given in (Garey and Johnson, 1979; p.257): the latter allows the number of colours available to form part of the instance; Levin's proof deals with tiles formed using a finite set of colours. The present authors can demonstrate a weak phase-transition in probability for Garey and Johnson's variant, however, this takes effect only as the number of available colours increases.

Problem 2: The obvious candidate is the property of being optimally edge-colourable, i.e. choose Π as

$$\{\ G(V,E) : G\ has\ chromatic\ index\ \Delta(G)\ \}$$

($\Delta(G)$ denoting the maximum degree of G).

A 'parity' type problem that is more 'natural' than the OE-$3COL$ example above, and for which it is unclear whether the device used there can be applied is OMC, i.e.

$$\{\ G(V,E) : The\ size\ of\ the\ largest\ clique\ in\ G\ is\ odd\ \}$$

Note, however, $OMC \in NP \cap co-NP$

6. Conclusion

In this paper we have proposed rigorous formulations of the statement 'f has a phase-transition' wherein it is contended that such assertions are only meaningful in the context of specific parameters of instances of f - i.e. the size and weight parameters σ, τ - and with respect to suitably presented measures. Our formulation makes a clear separation between purely combinatorial properties, such as the probability of an instance having a solution, and complexity-theoretic measures such as the run-time of algorithms to solve f. Within the framework developed, a number of results have been proved: it has been shown that the phase-transition in probability for 3-*SAT* does not imply that the run-time of search methods is polynomial for instances in the 'easy' regions; furthermore, for any method with probable polynomial run-time in some region there must be a superpolynomial number of 'hard' instances within this region. A number of contrasting results concerning phase-transitions in probability have been obtained although several interesting questions remain to be answered.

7. References

B. Bollobás and A. Thomason. Threshold Functions. *Combinatorica*, 7, (1987), 35-38

P. Cheeseman, B. Kanefsky, and W. Taylor. Where the really hard problems are. *Proc. of the 12th IJCAI*, (1991), 331-337

V. Chvátal and B. Reed. Mick gets some (the odds are on his side), *Proc. 33rd IEEE FOCS*, (1992), 620-627

M. Davis and H. Putnam. A computing procedure for quantification theory. *Jnl. of the ACM*, 7, (1960), 201-215

P. Erdös and A. Rényi. On Random Graphs I. *Pub. Math.*, 6, (1959), 290-297

E. Friedgut and G. Kalai. Every monotone graph property has a sharp threshold. *Proc. Amer. Mathl. Soc.*, 124, (1996), 2993-3002

J.W. Freeman. Hard random 3-SAT problems and the Davis-Putnam procedure. *Artificial Intelligence*, 81, (1996), 183-198

M.R. Garey and D.S. Johnson. Computers and Intractability. Freeman (1979)

I. Gent, and T. Walsh. The TSP phase transition. *Artificial Intelligence*, 81, (1996), 59-80

I. Gent, E. MacIntyre, P. Prosser, and T. Walsh. The constrainedness of search. *Proc. 13th National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, AAAI Press/ MIT Press, (1996), 246-252,

T. Hogg, B.A. Huberman, and C. Williams. Phase transitions and the search problem. *Artificial Intelligence*, 81, (1996), 1-15

T. Hogg and C. Williams. The hardest constraint problems: a double phase transition. *Artificial Intelligence*, 69, (1994), 359-377

S. Kirkpatrick and B. Selman. Critical behavior in the satisfiability of random Boolean expressions. *Science*, 264, (1994), 1297-1301

L.M. Kirousis, E.Kranakis, and D. Krizanc. Approximating the unsatisfiability threshold of random formulas. *Proc. 4th Annual European Symposium on Algorithms, ESA '96*, (September 1996)

A.D. Koršunov. Solution of a problem of Erdös and Rényi on Hamiltonian cycles in nonoriented graphs. *Soviet Math. Doklady*, 17, (1976), 760-764

L. Levin. Average case complete problems. *SIAM J. Comput.*, 15, (1986), 285-286

E. Koutsapias and C.H. Papadimitriou. On the greedy heuristic for satisfiability. *Inf. Proc. Letters*, 43, (1992), 53-55

D.G. Mitchell, B. Selman, and H.J. Levesque. Hard and easy distributions of SAT problems. *Proc. AAAI-92*, AAAI Press/ MIT Press, (1992), 459-465

L.J. Stockmeyer and A. Meyer. Word problems requiring exponential time. *Proc. 5th ACM Symposium on Theory of Computing*, (1973), 1-9

H. Wilf. Some examples of combinatorial averaging. *Amer. Math. Monthly*, 92, (1985), 250-261