Phase Transitions of Dominating Clique Problem and Their Implications to Heuristics in Satisfiability Search

Joseph Culberson, Yong Gao, Călin Anton
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada T6G 2E8
{joe,ygao,asy}@cs.ualberta.ca

Abstract

We study a monotone NP decision problem, the dominating clique problem, whose phase transition occurs at a very dense stage of the random graph evolution process. We establish the exact threshold of the phase transition and propose an efficient search algorithm that runs in super-polynomial time with high probability. Our empirical studies reveal two even more intriguing phenomena in its typical-case complexity: (1) the problem is "uniformly hard" with a tiny runtime variance on negative instances. (2) Our algorithm and its CNF-tailored implementation, outperform several SAT solvers by a huge margin on dominating cliques and some other SAT problems with similar structures.

1 Introduction

In this paper, we study the phase transition behavior of the dominating clique problem (DOMC), an NP-complete decision problem which for a given graph, asks if there is a subset of vertices that induces a clique and is a dominating set. Our interest in DOMC stems from an initial observation that its phase transition should occur when the random graph has a constant edge probability (i.e., has on average $\Omega(n^2)$ edges). This distinguishes DOMC from many other NP-complete problems previously studied from a phase transition perspective; Well-studied problems such as random SAT, random graph coloring, and Hamiltonian Cycle all have a phase transition when the underlying random graphs are sparse. For random SAT and graph coloring, it is known that randomly-generated instances become typically easy when the density parameter increases as a certain function of the problem size [Franco and Gelder, 2003; Krivelevich, 2002]. For Hamiltonian Cycle, whose edges-vertices ratio threshold is $\Omega(\log(n))$, instances randomly generated at the phase transition are already typically easy [Vandegriend and Culberson, 1998].

Another unique feature is that DOMC is a monotone decision problem without any extra parameter other than the problem size and edge density. This is in contrast to other similar partitioning and covering problems studied in the phase transition literature, such as the vertex cover [Hartmann and Weigt, 2003], in which an additional size parameter has to be used as an input.

There has also been recent interest in the study of the typical-case behavior of random problems with some deterministic structures, hoping that these problems might be able to capture more characteristics in real-world applications. Among others, the Quasigroup completion problem [Gomes and Shmoys, 2002], problems defined on the small-world graphs [Walsh, 1999], and problems obtained by "morphing" different problems [Gent *et al.*, 1999] have been proposed as alternatives to random problems with a uniform distribution. The dominating clique problem on random graphs and its CNF-encoding provide another type of distribution with (in some sense) extreme characteristics.

In Section 2, we establish the exact threshold of the phase transition of the dominating clique problem in random graphs. In Section 3, we present a backtracking algorithm for DOMC which will be used in our empirical investigation. In Section 4, we report two sets of interesting observations from our empirical investigation. We conclude in Section 5.

2 Dominating Clique Problem and its Phase Transitions

Consider a graph $G(V, E)$. A subset of vertices $V' \subset V$ is called a *dominating clique* of $G(V, E)$ if V' induces a clique and for any $v \in V \setminus V'$, there is a vertex $u \in V'$ such that $(u, v) \in E$. The dominating clique problem is the one that asks the question whether a given graph has a dominating clique.

The property of having a dominating clique is monotone and has a phase transition when the random graph is at the dense stage of the graph evolution process. The following theorem shows the phase transition has an exact threshold $p^* = \frac{3-\sqrt{5}}{2}$ for a random graph $G(n, p)$ where each of the $n(n-1)/2$ potential edges appears in the graph with probability p.

Theorem 2.1.

$$\lim_n Pr\{G(n,p) \textit{ has a dominating clique }\} = \begin{cases} 0, & \textit{if } p < \frac{3-\sqrt{5}}{2}; \\ 1, & \textit{if } p > \frac{3-\sqrt{5}}{2}. \end{cases} \quad (2.1)$$

To simplify the notations, we will write $b = \frac{1}{p}$. Let $DC(n,r)$ be the number of dominating cliques of size r. The expectation of $DC(n,r)$ is

$$E[DC(n,r)] = \binom{n}{r} p^{\binom{r}{2}} \left(1-(1-p)^r\right)^{n-r}. \quad (2.2)$$

By Stirling's formula, we have

$$E[DC(n,r)] \sim \frac{1}{\sqrt{2\pi r}} \left(\frac{enp^{(r-1)/2}}{r}\right)^r \left(1-(1-p)^r\right)^{n-r}. \quad (2.3)$$

Lemma 2.1. *If the edge probability p is such that $\log_b(b-1) > \frac{1}{2}$, then*

$$\lim_n \sum_{r=1}^{n} E[DC(n,r)] = 0. \quad (2.4)$$

Proof. Let

$$A(n) = \sum_{r=1}^{2\log_b n} \binom{n}{r} p^{\binom{r}{2}} \left(1-(1-p)^r\right)^{n-r}$$

and

$$B(n) = \sum_{r>2\log_b n} \binom{n}{r} p^{\binom{r}{2}} \left(1-(1-p)^r\right)^{n-r}$$

Based on the bound $\binom{n}{r} \le (\frac{en}{r})^r$, $B(n)$ can be estimated as follows:

$$\begin{aligned} B(n) &= \sum_{r>2\log_b n} \binom{n}{r} p^{\binom{r}{2}} \left(1-(1-p)^r\right)^{n-r} \\ &\le \sum_{r>2\log_b n} (\frac{en}{r})^r p^{\binom{r}{2}} \\ &= \sum_{r>2\log_b n} \left(\frac{\sqrt{b}en}{rb^{\frac{r}{2}}}\right)^r \\ &\le \sum_{r>2\log_b n} \left(\frac{\sqrt{b}e}{r}\right)^r \end{aligned}$$

Since $\frac{\sqrt{b}e}{r} < 1$ if n is sufficiently large, we have $\lim_n B(n) = 0$.

For $A(n)$, we have

$$\begin{aligned} A(n) &= \sum_{r=1}^{2\log_b n} \binom{n}{r} p^{\binom{r}{2}} \left(1-(1-p)^r\right)^{n-r} \\ &\le \sum_{r=1}^{2\log_b n} n^r \left(1-(\frac{b-1}{b})^r\right)^{n-r} \\ &\le (2\log_b n) e^{2\ln n \log_b n} \left(1-(\frac{b-1}{b})^{2\log_b n}\right)^{n-2\log_b n} \\ &\sim (2\log_b n) e^{2\ln n\log_b n} e^{-\left(\frac{b-1}{b}\right)^{2\log_b n}(n-2\log_b n)} \\ &= (2\log_b n) e^{2\ln n\log_b n} e^{-\frac{1}{n^2}n^{2\log_b(b-1)}(n-2\log_b n)} \\ &\sim (2\log_b n) e^{2\ln n\log_b n} e^{-n^{2\log_b(b-1)-1}} \end{aligned}$$

Therefore, we have $\lim_n A(n) = 0$ if $\log_b(b-1) > \frac{1}{2}$. □

Solving the inequality about b, we get $b > \frac{3+\sqrt{5}}{2}$, that is, $p < \frac{3-\sqrt{5}}{2} = 0.381966\cdots$. To prove the case of $p > \frac{3-\sqrt{5}}{2}$, i.e., $\log_b(b-1) < \frac{1}{2}$, we have to consider the variance of the number of dominating cliques in $G(n,p)$. Details will be given in the Appendix. Here, we only mention that the first crucial observation is that in this case, we have (A) $\frac{1}{1-\log_b(b-1)} < 2$ and (B) the expected number of dominating cliques of size $c\log_b n$ tends to infinity for any constant c such that $\frac{1}{1-\log_b(b-1)} < c < 2$.

Lemma 2.2. *Let c be a constant such that $\frac{1}{1-\log_b(b-1)} < c < 2$ and let $r = c\log_b n$. Then,*

$$\lim_n E[DC(n,r)] = +\infty.$$

Proof. By (2.3), we have

$$\begin{aligned} E[DC(n,r)] &\sim \frac{1}{\sqrt{2\pi r}} \left(\frac{\sqrt{b}en}{rb^{\frac{r}{2}}}\right)^r e^{-n^{c(\log_b(b-1)-1)+1}} \\ &= \frac{1}{\sqrt{2\pi r}} \left(\frac{en^{1-\frac{c}{2}}}{c\log_b n}\right)^{c\log_b n} e^{-n^{c(\log_b(b-1)-1)+1}}. \end{aligned}$$

The lemma follows since by assumption we have $c(\log_b(b-1)-1)+1 < 0$ and $1-\frac{c}{2} > 0$. □

3 An Algorithm for the Dominating Clique Problem

We present a brief over view of the elementary algorithm we developed for this problem.

We assume we are given a simple undirected graph $G = (V,E)$. The algorithm is a basic depth first backtrack algorithm that recursively constructs a potential dominating clique D. We call the set of vertices not adjacent to any vertex in D the uncovered set U. For a vertex to be a candidate for addition to D it must be adjacent to all vertices in D. We call this the selection set $S = \{x : \forall w \in D, \{x,w\} \in E\}$.

At each node in the backtrack tree, we select a pivot set $P \subseteq S$ such that if the current D can be extended to cover the graph at least one of the vertices must be in P. To this end, we first find a vertex $w \in U$ which has a minimum number of neighbors in S. Clearly either w or one of its neighbors in the current S must be in the final D. The effect of this is to greatly reduce the branching factor and the amount of redundant search.

For $A \subseteq V, w \in V$ we define the closed neighborhood of a vertex w in a subset A as $\widehat{N}(A,w) = \{x \in A : x = w \text{ or } \{x,w\} \in E\}$, and the open neighborhood $N(A,w) = \widehat{N}(A,w) \setminus \{w\}$. Initially, $U = S = V$ and $D = \emptyset$.

At first glance it appears we have eliminated the vertex of minimum degree, w, as a possible member of D. Suppose w is the only vertex in D, then since w is of minimum degree G is a clique and any vertex will do. Otherwise, on the initial step w will be in S' and so is available for inclusion later in the recursion. Correctness then follows by induction on the

```
DomClq(G, S, D, U) :: BOOLEAN
    If U = ∅ then
        DOMCLQ = D.
        return yes.
    endif
    Find w ∈ U such that |N(S, w)| = min_{ν∈U} |N(S, ν)|.
    P = N(S, w).
    if P = ∅ return no.
    S'' = S.
    while P ≠ ∅
        Select v ∈ P.
        D = D ∪ {v}.
        S' = N(S'', v).
        U' = U \ N̂(U, v).
        If DomClq(G, S', D, U') return yes.
        D = D \ {v}.
        S'' = S'' \ {v}.
        P = P \ {v}.
    endwhile
    return no;
end DomClq.
```

observations given above. Our assumption is that usually the vertex of minimum degree is not in D and so we may reduce the search cost by leaving it out of the initial P. Note that after the initial step S and U are disjoint.

In our program we made no effort to reorder the vertices of P for selection (first statement in the while loop). One obvious candidate is to select $v \in P$ to maximize $N(U, v)$. We would guess that in positive instances this would tend to find solutions quicker. However, the results for the negative instances in table 2 indicate this is unlikely to help in those cases, these are the most expensive and there would be a fair bit of overhead cost to this heuristic. We leave further investigation of selection heuristics as future work.

4 Empirical Studies

In this section, we discuss our empirical investigation on the dominating clique problem at phase transitions. In the first set of experiments, we study the threshold behavior and typical-case hardness of the dominating clique problem itself. In the second set of experiments, we compare the performance of our algorithm and some state-of-the-art SAT solvers.

4.1 Threshold and Uniformity of Hardness at Phase Transitions

Table 1 shows the preliminary result on empirical value of the threshold. Table 2 summarizes the typical case hardness at $p = 0.371$ for $n = 250 \ldots 1000$. The data indicate that the typical-case hardness at phase transitions increases at a super-polynomial rate. More interesting is the fact that for negative instances the variance in the number of the backtracks is tiny as compared to the total number. In other words, randomly-generated dominating clique instances just beyond the phase transition seem to be "uniformly" hard. This suggests that randomizing the selection within the pivot set P is unlikely to produce the heavy-tailed running time behavior frequently observed in many other problems [Gomes *et al.*, 2000].

n	Average	Std.	Min	Max
50	0.3725	0.0203	0.3176	0.4106
75	0.3698	0.0150	0.3258	0.4040
100	0.3685	0.0109	0.3358	0.3826
125	0.3700	0.0110	0.3406	0.3951
150	0.3663	0.0110	0.3316	0.3817
175	0.3660	0.0096	0.3376	0.3794
200	0.3689	0.0078	0.3502	0.3853
225	0.3680	0.0075	0.3448	0.3827
250	0.3669	0.0072	0.3483	0.3842
275	0.3669	0.0084	0.3445	0.3814
300	0.3674	0.0060	0.3523	0.3775
325	0.3685	0.0073	0.3434	0.3784
350	0.3671	0.0083	0.3407	0.3803
500	0.3675	0.0063	0.3357	0.3786
750	0.3680	0.0061	0.3568	0.3770

Table 1: The average threshold of Dominating Clique, expressed as a fraction of $\binom{n}{2}$ possible edges, for various n. Fifty instances at each n, except n=750 where only 8 instances completed. Std is the standard deviation, Min (Max) is the minimum (maximum) fraction encountered in the sample.

4.2 Significance of Using the Right Heuristics: How SAT solvers fail?

The dominating clique problem has a very natural and simple CNF encoding. To make the presentation easier, we adopt the convention that a clause can be viewed as a set of literals. Given a graph $G = G(V, E)$ with $V = \{v_1, \cdots, v_n\}$, the dominating clique problem can be encoded as follows.

1. There are n boolean variables $\{x_i, 1 \leq i \leq n\}$ where $x_i = 1$ signifies that the corresponding vertex v_i is in the dominating clique.
2. For each variable x_i, there is a clause $C_i = \{x_i\} \cup N(x_i)$, where $N(x_i)$ means the variables representing the vertices adjacent to v_i. This indicates that each vertex has to be in the dominating clique or have a neighbor in a dominating clique. These clauses will be called the *long-positive clauses*.
3. For each pair of variables (x_i, x_j) not in E, there is a clause $D_{i,j} = \{\overline{x}_i, \overline{x}_j\}$, indicating that at most one of the two non-adjacent vertices can be in a dominating clique. We will call these clauses the *short-negative clauses*.

Other encodings are also possible such as the support encoding [Gent, 2002] that maps maintaining arc-consistency of CSP algorithms to unit-propagation in SAT algorithms. But similar CNF encoding of the CSP representation for the dominating clique problem at the phase transitions will result in $\Omega(n^3)$ clauses of length $\Omega(n)$.

Backtrack nodes for YES instances				
N	Mean	Std.	Min.	Max
600	179031.3	127471.9	836.0	517596.0
700	347996.4	266229.1	460.0	1043853.0
800	658643.6	496335.5	1890.0	2199760.0
900	1484544.5	978648.8	100955.0	3992411.0
1000	2354966.6	1796775.3	9259.0	6586573.0
Backtrack nodes for NO instances				
N	Mean	Std.	Min.	Max
600	525205.8	7602.0	510705.0	537705.0
700	1121825.4	13552.0	1089553.0	1153289.0
800	2205960.3	19017.3	2166309.0	2254168.0
900	4063264.6	32241.0	3989194.0	4138323.0
1000	7103729.3	55926.8	6958133.0	7218638.0

Table 2: The number of backtrack nodes split on instances with and without a dominating clique. 100 instances at each N, $p = 0.371$. Note the remarkable consistency in the NO instances.

A SAT algorithm for the CNF-encoding of DOMC

We noticed that SATZ and ZChaff cannot solve any of the DOMC instances of size more than 600 in reasonable time. Therefore we translated our dominating clique algorithm to a SAT solver, called DCS, that solves the CNF-encoding of the dominating clique problem almost as efficiently as the orginal algorithm.

The solver does not use any of the current efficient data structures such as the technique of watched literals, nor does it use any advanced learning mechanism and heuristics, other than those directly translated from our dominating clique search algorithm. In the context of CNF formulas our algorithm reads as follows: (1) always pick an unassigned variable in the shortest remaining long-positive clause; (2) conduct a one-step unit propagation based on the information in short-negative clauses; and (3) stop as soon as all the long positive clauses are satisfied, or backtrack if a long-positive clause becomes empty. As has been discussed at the end of Section 3, our solver's performance may be further improved on satisfiable instances if we add the heuristic to select an unassigned variable that satisfies a maximum number of long-positive clauses.

N	DCS	BM	MQ	SZ	Solution Prob.
150	0.04	0.119	12.715	1.73	0.47
300	1.11	3.904	≥ 600	58.93	0.51

Table 3: Median running time (in seconds) of SAT solvers on random instances of the dominating clique problem. 100 instances for each N. The last column is the percentage of the satisfiable instances. Cutoff time is 600 seconds.

Results on the CNF encodings of DOMC and Subgraph Isomorphism

We have performed a series of experiments comparing the performance of our solver DCS and some state-of-the-art SAT solvers, including BerkMin(BM)[1], MarchEq(MQ)[2] and Satz(SZ).

First, we compared the performance of our solver DCS and other solvers on the CNF encoding of DOMC at phase transitions. We generated 100 random instances of the dominating clique problem at the edge probability $p = 0.368$ for $n = 150$ and 300. According to our experiment on the threshold, $p = 0.368$ is roughly the edge probability where 50 percent of the instances are satisfiable. The median running time (in seconds) of these solvers is summarized in Table 3.

Inspired by the efficiency of DCS on DOMC we investigated its performance on Subgraph Isomorphism (SGI) instances. Given two graphs $G(V_G, E_G)$ and $H(V_H, E_H)$, *the subgraph isomorphism problem* asks if there exists a subgraph H_1 of H such that G is isomorphic to H_1. A SGI instance can be naturally converted to SAT using an encoding similar to the one used for DOMC (direct encoding); the resulting SAT instance has $|V_G| \times |V_H|$ variables, $|V_G|$ long positive clauses and $O(|V_G|^2 \times |V_H|^2)$ short-negative clauses. SAT encodings of random SGI (SERSGI) are difficult for (deterministic) SAT solvers.[3]

We generated 5 sets of SERSGI instances with 528 variables. Each set consisted of 50 instances generated around the cross over point of SGI. DCS consistently outperformed all the other solvers on each set (Table 4). DCS is one or two orders of magnitude faster than MarchEq and Satz. This is remarkable especially considering the simplicity of DCS. To further investigate the speed up factor we compared the performances of DCS and BerkMin on the SERSGI instances used in the 2004 SAT competition (Table 5). With a single exception, DCS performance was comparable or better than that of BerkMin. Thus we conclude that DCS outperforms state of the art solvers on SERSGI instances.

It may be possible to extend this conclusion to other classes of SAT instances which have only long-positive and short negative clauses, but more work needs to be done to see what are the structural properties that DCS can exploit. However, we do not expect that DCS will work efficiently on any type of CNF formulas with a long-short clause structure. We noticed that on random k-SAT instances, it is at least one order of magnitude slower than Satz. We consider this behavior an evidence against the idea of an efficient universal sat solver that does not consider any problem structure.

5 Conclusions

Most of the well-investigated NP-complete problems in the literature have a phase transition when the underlying graph is very sparse. There is also evidence that randomly-generated instances at the phase transition are typically not hard if the

[1] BerkMin is the predecessor of Forklift the winner of the Industrial Category at the 2003 SAT competition. Our experience indicates that Zchaff (winner of 2004) is not efficient for this type of instances

[2] Winner of the Hand Crafted Category at the 2004 SAT competition

[3] At the 2004 SAT competition[LeBerre and Simon, 2004], no deterministic solver could solve any SERSGI instance.

	Set 1	Set 2	Set 3	Set 4	Set 5
% Solved					
DCS	100	100	100	96	100
BM	100	90	80	58	44
MQ	94	66	36	12	2
SZ	50	22	2	14	2
Median Solution Time (sec)					
DCS	14.48	19.14	54.92	131.23	135.50
BM	30.8	97.03	250.00	919.28	≥ 1200
MQ	233.74	905.15	≥ 1200	$\geq$1200	≥ 1200
SZ	1111.75	≥ 1200	≥ 1200	$\geq$1200	≥ 1200
Mean Solution Time (sec)					
DCS	42.09	44.21	80.98	209.07	240.47
BM	90.47	283.00	479.91	-	-
MQ	383.20	-	-	-	-
SZ	-	-	-	-	-

Table 4: Performances of DCS, BerkMin, MarchEq and Satz on 5 sets of SERSGI instances with 528 variables. 50 instances in each set. Cutoff time 1200 seconds.

Inst#	776	770	772	774	767	765	767
DCS	79.1	73.7	8.7	86.3	128.0	121.4	18.35
BM	526.4	33.3	58.5	79.9	2930.9	366.9	28.21

Table 5: Running time (minutes) of DCS and BerkMin on SERSGI instances used in 2004 SAT competition.

threshold can be accurately established such as the Hamiltonian Cycle problem. DOMC studied in this paper is distinguished from a phase transition and typical-case complexity perspective. On the one hand, it has an exact threshold at a very dense stage of the random graph evolution process. Our empirical results, on the other hand, indicate that randomly-generated instances are still typically (and uniformly for negative instances) hard. As has been demonstrated by our SAT solver's outstanding performance advantage on the CNF-encoding of DOMC and other problems with similar structures, DOMC might serve as a prototypical problem to further investigate the relationship among the problem structure, the encoding efficiency, and the use of relevant heuristics in satisfiability search.

Appendix: proof of Theorem 2.1

To estimate the second moment of the number of dominating cliques of size r, we need the following lemma on the probability that two vertex subsets both dominate the rest of the vertices in a random graph.

Lemma 5.1. *Let U and W be two vertex subsets of size r such that $|U \cap W| = l$ and let $P(l)$ be the probability that U and W both dominate all the vertices outside of $U \cup W$. Then,*

1.
$$P(l) = \left[1 - 2(1-p)^r + (1-p)^{2r-l}\right]^{n-(2r-l)} \quad (5.5)$$

2. *Let $Q(n,r) = (1-(1-p)^r)^{2n-2r}$. We have*

   (a) $\lim_n \frac{P(0)}{Q(n,r)} = 1$*; and*

   (b) There exists a constant C^* and N^* such that if $n > N^*$, $\frac{P(l)}{Q(n,r)} < C^*$ for all $1 \leq l \leq r$.

Proof. (1). Let v be a vertex outside of $U \cup W$. The probability that v is not connected to any vertices in $U \cap W$ is $(1-p)^l$. The probability that v is dominated by both $U \setminus W$ and $W \setminus U$ is
$$(1-(1-p)^{r-l})^2.$$
Thus, $P(l)$ is
$$\begin{aligned}
&\left[(1-(1-p)^l) + (1-p)^l(1-(1-p)^{r-l})^2\right]^{n-(2r-l)} \\
&= [1-(1-p)^l + (1-p)^l(1-2(1-p)^{r-l} \\
&\quad +(1-p)^{2(r-l)})]^{n-(2r-l)} \\
&= \left[1-2(1-p)^r + (1-p)^{2r-l}\right]^{n-(2r-l)}
\end{aligned}$$

(2). Since $log_b(b-1) - 1 < 0$, we have
$$\begin{aligned}
\lim_n \frac{P(0)}{Q(n,r)} &= \lim_n \frac{(1-(1-p)^r)^{2(n-2r)}}{(1-(1-p)^r)^{2n-2r}} \\
&= \lim_n (1-(1-p)^r)^{-2r} \\
&= \lim_n e^{-(1-p)^r(-2r)} \\
&= \lim_n e^{2rn^{c(log_b(b-1)-1)}} \\
&= 1.
\end{aligned}$$

For $1 \leq l \leq r$, we have
$$\begin{aligned}
\frac{P(l)}{Q(n,r)} &= \frac{\left[1-2(1-p)^r + (1-p)^{2r-l}\right]^{n-(2r-l)}}{(1-(1-p)^r)^{2n-2r}} \\
&\leq \frac{\left[1-2(1-p)^r + (1-p)^{2r-l}\right]^{n-2r}}{(1-(1-p)^r)^{2n-2r}} \\
&\leq \frac{[1-2(1-p)^r + (1-p)^r]^{n-2r}}{(1-(1-p)^r)^{2n-2r}} \\
&= \left(\frac{1}{1-(1-p)^r}\right)^n
\end{aligned}$$

Since
$$\begin{aligned}
\lim_n \left(\frac{1}{1-(1-p)^r}\right)^n &= \lim_n \left(1 + \frac{(1-p)^r}{1-(1-p)^r}\right)^n \\
&= \lim_n e^{n(1-p)^r \frac{1}{1-(1-p)^r}} \\
&= O(1) \lim_n e^{n^{c(log_b(b-1)-1)+1}}
\end{aligned}$$

and $c(log_b(b-1)-1)+1 < 0$, conclusion 2(a) follows. □

Proof of Theorem 2.1. The fact that
$$\lim_n Pr\{G(n,p) \text{ has a dominating clique }\} = 0$$
for $p < \frac{3-\sqrt{5}}{2}$ follows from Lemma 2.1 and Markov's inequality.

To prove that $\lim_n Pr\{G(n,p) \text{ has a dominating clique }\} = 1$ for $p > \frac{3-\sqrt{5}}{2}$, we will show that

$$\lim_n Pr\{DC(n,r) = 0\} = 0$$

where $r = c\log_b n$ and $\frac{1}{1-\log_b(b-1)} < c < 2$. From Lemma 2.2, we have

$$\lim_n E[DC(n,r)] = +\infty.$$

By Chebyshev's inequality, we have

$$Pr\{DC(n,r) = 0\} \leq \frac{E[DC^2(n,r)]}{E^2[DC(n,r)]} - 1.$$

The expectation of $DC^2(n,r)$ is

$$E[DC^2(n,r)] = \sum_{l=0}^{r} \binom{n}{r}\binom{r}{l}\binom{n-r}{r-l} p^{2\binom{r}{2}-\binom{l}{2}} P(l).$$

where $\binom{n}{r}\binom{r}{l}\binom{n-r}{r-l}$ is the total number of pairs of vertex subsets of size r that have l vertices in common, and $p^{2\binom{r}{2}-\binom{l}{2}}P(l)$ is the probability that such a pair of vertex subsets are both dominating cliques. Recall that

$$\begin{aligned} E^2[DC(n,r)] &= \left(\binom{n}{r} p^{\binom{r}{2}}(1-(1-p)^r)^{n-r}\right)^2 \\ &= \binom{n}{r}^2 p^{2\binom{r}{2}} Q(n,r). \end{aligned}$$

We write $\frac{E[DC^2(n,r)]}{E^2[DC(n,r)]}$ as

$$\begin{aligned} \frac{E[DC^2(n,r)]}{E^2[DC(n,r)]} &= \frac{\sum_{l=0}^{r}\binom{n}{r}\binom{r}{l}\binom{n-r}{r-l}p^{2\binom{r}{2}-\binom{l}{2}}P(l)}{\binom{n}{r}^2 p^{2\binom{r}{2}}Q(n,r)} \\ &= a_n + b_n, \end{aligned}$$

where

$$\begin{aligned} a_n &= \frac{\sum_{l=0}^{1}\binom{n}{r}\binom{r}{l}\binom{n-r}{r-l}p^{2\binom{r}{2}-\binom{l}{2}}P(l)}{\binom{n}{r}^2 p^{2\binom{r}{2}}Q(n,r)} \\ &= \binom{n}{r}^{-1}\left[\binom{n-r}{r}\frac{P(0)}{Q(n,r)} + r\binom{n-r}{r-1}\frac{P(1)}{Q(n,r)}\right]. \end{aligned}$$

and

$$b_n = \binom{n}{r}^{-1}\sum_{l=2}^{r}\binom{r}{l}\binom{n-r}{r-l}p^{-\binom{l}{2}}\frac{P(l)}{Q(n,r)}.$$

We will prove that $\lim_n a_n = 1$ and $\lim_n b_n = 0$. Since $\frac{\binom{n-r}{r}}{\binom{n}{r}} \to 1$ and $\frac{r\binom{n-r}{r-1}}{\binom{n}{r}} \to 0$, by Lemma 5.1(2.a and 2.b) we have

$$\lim_n a_n = 1.$$

From Lemma 5.1 (2.b), we have for $n > N^*$

$$\begin{aligned} b_n &\leq C^* \sum_{l=2}^{r}\binom{n}{r}^{-1}\binom{r}{l}\binom{n-r}{r-l}p^{-\binom{l}{2}} \\ &\leq O(1)\sum_{l=2}^{r}\frac{r^{2l}}{l!n^l}b^{\binom{l}{2}} \\ &= O(1)\sum_{l=2}^{r}\left(\frac{r^2 b^{l/2}}{n\sqrt{b}}\right)^l \end{aligned}$$

Since $r = c\log_b n$ with $c < 2$, we have

$$b_n \leq O(1) r\left(\frac{r^2}{n^{1-c/2}\sqrt{b}}\right)^r \to 0.$$

This completes the proof. □

References

[Franco and Gelder, 2003] J. Franco and A. V. Gelder. A perspective on certain polynomial-time solvable classes of satisfiability. *Discrete Applied Mathematics*, 125(2-3):177–214, 2003.

[Gent *et al.*, 1999] I. Gent, H. Hoos, P. Prosser, and T. Walsh. Morphing: Combining structure and randomness. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI'99)*, pages 654–660, Orlando, Florida, 1999.

[Gent, 2002] I. Gent. Arc consistency in SAT. In *Proceedings of the 15th European Conference on Artificial Intelligence (ECAI 2002)*, pages 121–125, 2002.

[Gomes and Shmoys, 2002] C. Gomes and D. Shmoys. Completing quasigroups or latin squares: A structured graph coloring problem. In *Proceedings of the Computational Symposium on Graph Coloring and Extensions*, 2002.

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

[Hartmann and Weigt, 2003] A. Hartmann and M. Weigt. Statistical mechanics of the vertex-cover problem. *Journal of Physics A: Mathematical and General (a special issue)*, 36(43):11069–11093, 2003.

[Krivelevich, 2002] M. Krivelevich. Coloring random graphs - an algorithmic perspective. In *Proceedings of the 2nd Colloquium on Mathematics and Computer Science (MathInfo'2002)*, pages 175–195. Birkhauser, 2002.

[LeBerre and Simon, 2004] D. LeBerre and L. Simon. SAT competition - webpage, 2004. Available from: satlive.org/SATCompetition/2004.

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

[Walsh, 1999] T. Walsh. Search in a small world. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 1172–1177, 1999.