SAT-Variable Complexity of Hard Combinatorial Problems

Kazuo IWAMA *and* Shuichi MIYAZAKI
Department of Computer Science and Communication Engineering
Kyushu University, Hakozaki, Higashi-ku, Fukuoka 812, Japan
iwama@csce.kyushu-u.ac.jp

Abstract. This paper discusses polynomial-time reductions from Hamiltonian Circuit (HC), k-Vertex Coloring (k-VC), and k-Clique Problems to Satisfiability Problem (SAT) which are efficient in the number of Boolean variables needed in SAT. We first present a basic type of reductions that need $(n-1)\log(n-1)$, $(n-1)\log k$, and $k\log n$ variables for HC, k-VC and k-Clique, respectively. Several heuristics can reduce the number of variables. Some of them achieve: $(n-I-1)\log(n-1)+\Sigma\log d_i$ for HC (I is the size of any independent set of vertices v_i's whose degree is d_i's), and $\log n+(k-1)\log D$ for k-Clique (D is the kth largest degree of the graph).

Recent revolutionary progress in SAT algorithms will make it increasingly reasonable to solve (hard) combinatorial problems after reducing them to SAT. Efficiency in the above sense apparently plays a key role in this approach. From a different viewpoint, the number of variables can act as a complexity measure for the original problems, if the reduction is sufficiently efficient. The merits of heuristics can also be evaluated by (the reduction of) this complexity.

Keyword Codes: F.2.2; G.2.1; G.2.2
Keywords: Nonnumerical Algorithms and Problems; Combinatorics; Graph Theory

1. Introduction

Finding of the usefulness of the local search method [KP92, SLM92, MI92] brought a great impact to the world of SAT algorithms. Currently it is claimed that the local search method can find a satisfying truth assignment of CNF equations with up to 10^4 variables under certain conditions [SK93]. That means it will be increasingly reasonable to solve other (hard) combinatorial problems by reducing (in polynomial time) them to SAT. For this strategy to work, however, it is apparently significant that the reduction is efficient in the sense that the size of instances does not increase much. For example, if one relied upon a careless reduction that changes some graph of n vertices to a predicate of n^2 variables, then the merit (even if it is vast) of SAT algorithms would be easily canceled. (Recall that many NP-completeness proofs do use such "inefficient" reductions which increases the size of instances by square or even cube.)

However, many particular problems appear to have much more efficient reductions. For example, the k-Vertex Color (k-VC) Problem can be reduced to SAT using $(n-1)\log k$ variables or less (n is the number of vertices of the given graph, $\log n$ always mean $\lceil \log_2 n \rceil$ in this paper, see Section 4). Suppose that SAT can be "solved" up to 10^4 variables. (There are already preliminary experimental results in [SK93] and by ourselves that suggests the reality of this estimation.) Then this reduction allows us to solve the k-VC problem with up to some 10^3 vertices. This may be very surprising, since it is commonly understood [J93] that even 10^2 variables are very hard to cope with under the current technique for the k-VC problem. Thus the efficient reduction can create the new possibility that the solvable size of hard combinatorial problems increases significantly.

Another purpose of this paper is to claim that the number of SAT variables, if the reduction is reasonably efficient, can be a measure of the complexity of the original problem. We shall call this measure the SAT variable (SV) complexity.

(1) As will be mentioned in Sections 3 and 5, the Hamiltonian Circuit (HC) and the k-Clique problems can also be reduced to SAT with $(n-1)\log(n-1)$ and $k \log n$ variables, respectively. Including the case of HC too, those values well coincide with (the logarithm of) their intuitive worst-case complexities, for there are $(n-1)!$ different vertex orders as possible Hamiltonian Circuits and ${}_nC_k$ vertex subsets for k-Clique.

(2) Note that there is no obvious way to reduce the SV complexity of HC from $(n-1)\log(n-1)$ to even $(n-1)\log(n-1)-1$. Thus the SV complexity is accurate and hence we can expect that it is sensitive enough to measure the effect of usual heuristics that certainly work practically but are not easy to be analyzed mathematically. For example, consider the following (simple) heuristic for k-VC: Look at a vertex v and $v_1, v_2, \cdots, v_l$ adjacent to v. Then the number of colors those vertices $v_1, v_2, \cdots, v_l$ can use is not k but at most $k-1$. It is far from easy to analyze how much the original search space (k^n) can be reduced by this heuristic. However, we can show that the SV complexity decreases by this heuristic from $(n-1)\log k$ to $(n-I-1)\log k$ if k is a constant, where I is the size of any independent set that can be computed in polynomial time. It is also shown that there do exist heuristics which can decrease the SV complexity of HC by at least one.

There is a fairly large literature discussing (average in many cases) complexities of concrete NP problems (see e.g., [MS92, DF89, BCLS87]), but have been relatively few papers [L86, DCGL92] that investigate the relation among problems, after the explosion of NP-completeness papers. However, we believe that the reduction of one problem to another often gives us a new insight on the original problem, which possibly contributes to developing fast algorithms. For example, almost all existing graph-color algorithms essentially depend on Kucera's method [L86]. However, if it is converted to SAT and the local search is applied to the SAT, then the overall algorithm, when considered in the original graph world, must be totally different from Kucera's method.

2. Preliminaries

In this paper, the following four problems are discussed. They are so-called decision problems (the answer to each instance is Yes or No). *Hamiltonian Circuit Problem* (*HC*): For a graph $G=(V,E)$, HC asks if G has an ordering $\langle v_0, v_1, \cdots, v_{n-1}\rangle$ of the vertices of G, where $n=|V|$, such that $(v_{n-1}, v_0) \in E$ and $(v_i, v_{i+1}) \in E$ for $0 \le i < n-1$. *k-Vertex Color Problem* (*k-VC*): For a graph $G=(V,E)$ and a positive integer k, it asks if there exists a coloring function f: $V \to \{0,1,\cdots,k-1\}$ such that $f(u) \neq f(v)$ whenever $(u,v) \in E$. *k-Clique*: For a graph $G=(V,E)$ and a positive integer k, it asks if G contain a clique of size k or more, that is, a subset $V' \subseteq V$ such that $|V'| \ge k$ and every two vertices in V' are joined by an edge in E. *Satisfiability* (*SAT*): It asks whether there is a satisfying truth assignment for a given CNF predicate F, i.e., a collection of clauses.

In this paper, a reduction M always means a many-one reduction to SAT from another problem. M must run in deterministic polynomial time but what kind of polynomial (e.g., n^2) does not matter. Suppose that M reduces a problem P to SAT in such a way that any P's instance of size n can be translated into a CNF predicate of at most $f(n)$ variables. (Again we do not care the number of clauses, which must be polynomial since M runs in polynomial time.) Then M is said to be $f(n)$ *SAT-variable* (*SV*) *bounded*. We also say that the *SV complexity* of the problem P is $\tilde{O}(f(n))$. $\tilde{O}$ means an upperbound but, unlike the usual Big-O, we do not neglect a constant factor or lower-order terms. For example, $\tilde{O}(f(n)) \neq \tilde{O}(f(n)-1)$. Since the SV complexity is only the complexity measure discussed in this paper, we often use simplified description such as "an $\tilde{O}(f(n))$ reduction".

3. Reductions from HC

3.1. Basic Reduction

We first present an $\tilde{O}((n-1)\log(n-1))$ reduction. Let $G=(V,E)$ be a graph of n vertices $v_0, v_1, \cdots, v_{n-1}$. For each vertex v_i ($i \neq 0$), we prepare $N = \log(n-1)$ variables $x_{i,0}, x_{i,1}, \cdots, x_{i,N-1}$. The total number of variables is $(n-1)\log(n-1)$. The basic idea of constructing a CNF predicate F (i.e., a set of clauses) is as follows: Suppose that the sequence

of vertices $v_0, v_{i_1}, v_{i_2}, \cdots, v_{i_j}, \cdots, v_{i_{n-1}}, v_0$ is a Hamiltonian Circuit of G. Then $\boldsymbol{X}_{i_1} = 0, \boldsymbol{X}_{i_2} = 1, \cdots, \boldsymbol{X}_{i_j} = j-1, \cdots, \boldsymbol{X}_{i_{n-1}} = n-2$ is a satisfiable assignment of F, where $\boldsymbol{X}_{i_j} = j-1$ means to assign the binary representation of value $j-1$ to $(x_{i_j,N-1}, x_{i_j,N-2}, \cdots, x_{i_j,0})$. Also, let $C((x_{i,N-1}, \cdots, x_{i,0}) = l)$ denote the (single) clause that becomes 0 if $(x_{i,N-1}, \cdots, x_{i,0}) = l$. For example, $C((x_{i,3}, x_{i,2}, x_{i,1}, x_{i,0}) = 5)$ is $(x_{i,3} + \overline{x_{i,2}} + x_{i,1} + \overline{x_{i,0}})$.

Reduction HC-I.

Step1: For each i, $1 \leq i \leq N-1$, construct the clauses that become 0 (false) if $(x_{i,N-1}, \cdots, x_{i,0}) > n-2$ that means the truth assignment to $(x_{i,N-1}, \cdots, x_{i,0})$, if it is viewed as a binary number, is grater than $n-2$. (The similar notations will also be used below.) For example, when $N = 3$ and $n-2 = 1010$, $(\overline{x_{i_3}} + \overline{x_{i_2}})(\overline{x_{i_3}} + x_{i_2} + \overline{x_{i_1}} + \overline{x_{i_0}})$ are those clauses.

Step2: For each i, j, $1 \leq i < j \leq n-1$, construct the clauses that becomes 0 if $(x_{i,N-1}, \cdots, x_{i,0}) = (x_{j,N-1}, \cdots, x_{j,0})$. A simple method is to prepare 2^N clauses for each i, j, namely, for each l, $0 \leq l \leq 2^N - 1$, construct (single) clause $C((x_{i,N-1}, \cdots, x_{i,0}) = l) + C((x_{j,N-1}, \cdots, x_{j,0}) = l)$

Step3: For each i, j, $1 \leq i < j \leq n-1$, such that v_i and v_j are not adjacent, construct the clauses that become 0 if $(x_{i,N-1}, \cdots, x_{i,0}) = (x_{j,N-1}, \cdots, x_{j,0}) - 1$ or $(x_{i,N-1}, \cdots, x_{i,0}) = (x_{j,N-1}, \cdots, x_{j,0}) + 1$. Use the same technique as Step2.

Step4: For each i such that v_i is not adjacent to v_0, construct two clauses $C((x_{i,N-1}, \cdots, x_{i,0}) = n-2)$ and $C((x_{i,N-1}, \cdots, x_{i,0}) = 0)$.

Theorem 1. *There is a reduction from HC whose SV complexity is* $\widetilde{O}((n-1)\log(n-1))$.

3.2. $\widetilde{O}((n-I-1)\log(n-1) + \Sigma \log d_i)$ Reduction

It should be noted that there are no obvious ways of decreasing the SV complexity of HC-I by even one. (For example, suppose that one could always find in polynomial time a vertex v such that if G has a Hamiltonian Circuit then it has a Hamiltonian Circuit in which v appears in the first half. Then the number of necessary variables for this v would be $N-1$ instead of N. Unfortunately, no such properties are known for general graphs.)

In this section, we shall show more sophisticated reduction than HC-I which can save one variable at worst and more on average. The basic idea is as follows: It is a well-known fact (e.g., [B73]) that if every vertex of G has degree at least $n/2$ then G has a Hamiltonian Circuit. This condition can obviously be checked in polynomial time. Hence we can assume that G includes at least one vertex v of degree at most $n/2 - 1$. To this vertex v, we prepare not N variables but $\log d_v$ variables (d_v is v's degree, $\log(d_v) \leq N-1$ since $d_v < n/2$). Now we consider that the value assigned to these variables (say, h, $0 \leq h \leq d_v - 1$) means that the previous vertex of v in the Hamiltonian Circuit is the vertex which is connected to v by the hth edge out of the d_v one's. (We must set up a fixed order among the d_v vertices adjacent to v in advance.) The SV complexity of the following HC-II is thus $(n-2)\log(n-1) + \log D$ where D is the minimum degree of G. Prepared variables are the same as HC-I (i.e., $x_{i,0}, \cdots, x_{i,N-1}$) except for the minimum-degree vertex v for which $y_0, y_1, \cdots, y_{N'-1}$ ($N' = \log d_v$) are prepared.

Reduction HC-II.

Step1: Similarly as HC-I, construct clauses to maintain fundamental restrictions for variables, such as (i) $(x_{i,N-1}, \cdots, x_{i,0}) \leq n-2$, (ii) $(y_{N'-1}, \cdots, y_0) \leq d_v - 1$ and (iii) $(x_{i,N-1}, \cdots, x_{i,0}) \neq (x_{j,N-1}, \cdots, x_{j,0})$ if $i \neq j$.

Step2: The same restriction as Step1-(iii) for the designated vertex v, namely if v is at the pth position in the Hamiltonian Circuit, then no other vertex can be at the same pth. For each i, j, l and p such that $0 \leq i \leq n-2$, v_j is adjacent to v by the lth edge, and $0 \leq p \leq n-2$, construct single clause

$$C((x_{i,N-1}, \cdots, x_{i,0}) = p) + C((y_{N'-1}, \cdots, y_0) = l) + C((x_{j,N-1}, \cdots, x_{j,0}) = p-1).$$

Step3: The same as Step3 of HC-I. If v_i and v_j are not adjacent, $(x_{i,N-1}, \cdots, x_{i,0}) \neq (x_{j,N-1}, \cdots, x_{j,0}) \pm 1$.

Step4: The same restriction as Step3 for the designated vertex v. For each i, j, l and p such that v_i and v are not adjacent, v_j is adjacent to v by the lth edge, and $0 \leq p \leq n-2$, construct

single clause

$$C((x_{i,N-1},\cdots,x_{i,0})=p)+C((y_{N'-1},\cdots,y_0)=l)+C((x_{j,N-1},\cdots,x_{j,0})=p-2).$$

Step5: Handle the special case for Step3 and Step4, namely, when $v_i = v_0$. Omitted.

Although details are omitted, HC-II can be extended so as to work for the existence of two or more designated vertices u_i for which $y_{i,\log d_i-1}, y_{i,\log d_i-2},\cdots,y_{i,0}$ are prepared (instead of N variables) if any two of them are not adjacent. The following theorem thus follows:

Theorem 2. *Suppose that $\{u_1,u_2,\cdots,u_I\}$ is any independent set of vertices. Then there is a reduction from HC whose SV complexity is $\widetilde{O}((n-I-1)\log(n-1)+\Sigma\log d_i)$, where d_i is the degree of vertex u_i.*

Remark 1. *Suppose that the degree of G is small, say at most four in all vertices. Then one might think that $2(n-1)$ variables are enough since, from each vertex two variables are enough to specify which edge out of at most four the Hamiltonian Circuit goes through. This is actually not true: It appears to be impossible to write appropriate predicate of polynomial length using this few variables. On the other hand, if one uses more (say, n^2) variables then it becomes easier to construct the predicates whose length is likely to decrease. However, as mentioned earlier, increasing the number of variables is considered to be worse than anything.*

4. Reductions from k-VC

4.1. Basic Reduction

The SV complexity of the following basic reduction is $\widetilde{O}((n-1)\log k)$. Similarly as before, for each vertex $v_i (i\neq 0)$, we prepare $N=\log k$ variables $x_{i,0},x_{i,1},\cdots,x_{i,N-1}$. The total number of variables is $(n-1)\log k$. Suppose the solution of k-VC is that $v_i (i\neq 0)$ is colored by C_i. (We fix the color of v_0 to be color 0.) Then we construct a CNF predicate F so that $\boldsymbol{X}_1=C_1,\boldsymbol{X}_2=C_2,\cdots,\boldsymbol{X}_i=C_i,\cdots,\boldsymbol{X}_{n-1}=C_{n-1}$ will be a satisfiable assignment of F.

Reduction kVC-I.

Step1: For each i, $0\leq i\leq n-1$, construct the clauses that become 0 if $(x_{i,N-1},\cdots,x_{i,0})>k-1$.

Step2: For each i,j such that v_i and v_j are adjacent, construct a clause that becomes 0 if $(x_{i,N-1},\cdots,x_{i,0})=(x_{j,N-1},\cdots,x_{j,0})$. Namely, for each i,j and p such that v_i and v_j are adjacent, and $0\leq p\leq k-1$, construct a clause

$$C((x_{i,N-1},\cdots,x_{i,0})=p)+C((x_{j,N-1},\cdots,x_{j,0})=p).$$

Step3: The special case for Step2, for $v_i=v_0$. For each i such that v_i is adjacent to v_0, construct a clause

$$C((x_{i,N-1},\cdots,x_{i,0})=0).$$

Theorem 3. *There is a reduction from k-VC whose SV complexity is $\widetilde{O}((n-1)\log k)$.*

4.2. $\widetilde{O}((n-I-1)\log k)$ Reduction

Consider a vertex v (a designated vertex, whose degree is l) and $u_0,u_1,\cdots,u_{l-1}$ adjacent to v. Then the number of colors $u_0,u_1,\cdots,u_{l-1}$ can use is at most $k-1$, because once all the k colors are used for $u_0,u_1,\cdots,u_{l-1}$, the vertex v is no longer colorable. Otherwise, i.e., if we can guarantee that $u_0,u_1,\cdots,u_{l-1}$ use at most $k-1$ colors, then the vertex v can use (at least one) remaining color. Thus we can save all the N variables that were prepared for v in kVC-I.

Reduction kVC-II.

Step1: Similarly as kVC-I, construct clauses which maintain the restriction $(x_{i,N-1},\cdots,x_{i,0})\leq k-1$.

Step2: Similarly as step2 of kVC-I, if v_i and v_j are adjacent, $(x_{i,N-1},\cdots,x_{i,0})\neq(x_{j,N-1},\cdots,x_{j,0})$.

Step3: For the designated vertex v (degree l), let $u_0,\cdots,u_{l-1}$ are adjacent to v. If $l<k$, we do not construct any clause. If $l\geq k$, for each combination of k vertices $(u_{i_0},u_{i_1},\cdots,u_{i_{k-1}})$ selected from $(u_0,\cdots,u_{l-1})$, and any permutation $\sigma_0,\sigma_1,\cdots,\sigma_{k-1}$ of $0,1,\cdots,k-1$, con-

struct a clause

$$C((x_{i_0,N-1},\cdots,x_{i_0,0})=\sigma_0)+\cdots+C((x_{i_{k-1},N-1},\cdots,x_{i_{k-1},0})=\sigma_{k-1}).$$

It should be noted that the number of clauses is polynomial if k is a constant.
Step4: Handle the special case for Step2 and Step3, namely, for $v_i = v_0$. Omitted.

kVC-II can be extended for two or more designated vertices v, for which no variables are prepared, if any two of them are not adjacent. So if we can find an independent set of size I in polynomial time, the SV complexity can be reduced to $\widetilde{O}((n-I-1)\log k)$.

Theorem 4. *There is a reduction from k-VC (k is constant) whose SV complexity is $\widetilde{O}((n-I-1)\log k)$, where I is the size of any independent set of the given graph obtained in polynomial time.*

Remark 2. *Apply the well known algorithm to obtain a maximal independent set for this purpose. Then for a random graph such that there exists an edge with probability p between every two vertices, it turns out to be $I \simeq \frac{\log n}{\log n+pn}n$. Namely, if the graph is sparse, for example, if pv (the average degree)$\simeq \log n$, then the complexity becomes roughly a half.*

4.3. $\widetilde{O}(2(n-1)-|S|)$ Reduction

In this section we introduce another heuristic for the case that k is constant. For simplicity, we assume that $k = 4$, i.e., the graph should be colored with four different colors. (Extension to the general k is not hard.) The basic idea is simple: Suppose that v_1, v_2 and v_3 constitute a subgraph of K_3. Then, if the colors of v_1 and v_2 are fixed to, say, color 0 and 1, then the possible colors for v_3 are not four but only two (colors 2 and 3). That means we need only one variable, instead of two, for v_3. (That variable=0 means the smaller-numbered color out of the two remaining ones.) It should be noted that if v_4 is also adjacent to v_1 and v_2, then we need only one variable for this v_4 also. We omit the polynomial-time reduction algorithm.

Theorem 5. *There is a reduction from 4-VC whose SV complexity is $\widetilde{O}(2(n-1)-|S|)$ where S is any set of vertices v such that v is adjacent to some vertices v_1 and v_2 in $V-S$ and $(v_1,v_2)\in E$.*

5. Reductions from k-Clique

5.1. Basic Reduction

The basic reduction needs $k\log n$ variables, which are divided into k sets of $N=\log n$ variables, namely, $x_{i,0}, x_{i,1},\cdots,x_{i,N-1} (0 \le i \le k-1)$. Then a CNF predicate F is naturally constructed so that if vertex $v_{i_0}, v_{i_1},\cdots,v_{i_{k-1}}$ constitute a k-clique, then $\boldsymbol{X}_0 = i_0, \boldsymbol{X}_1 = i_1,\cdots,\boldsymbol{X}_j = i_j,\cdots,\boldsymbol{X}_{k-1} = i_{k-1}$ will be a satisfying assignment of F.

Reduction kClique-I.
Step1: For each i $(0 \le i \le k-1)$, construct the clauses that become 0 if $(x_{i,N-1},\cdots,x_{i,0}) > n-1$
Step2: For each i, j, $0 \le i < j \le k-1$, construct the clauses that become 0 if $(x_{i,N-1},\cdots,x_{i,0}) = (x_{j,N-1},\cdots,x_{j,0})$
Step3: For each pair of vertices v_i, v_j such that v_i and v_j are not adjacent, and for each p, q, $0 \le p < q \le k-1$, construct a single clause $C((x_{p,N-1},\cdots,x_{p,0})=i)+C((x_{q,N-1},\cdots,x_{q,0})=j)$.

Theorem 6. *There is a reduction from k-Clique whose SV complexity is $\widetilde{O}(k\log n)$.*

5.2. $\widetilde{O}(\log n+(k-1)\log D)$ Reduction

There exists a more efficient reduction (kClique-II): It is obvious that all the $k-1$ vertices in k-clique are adjacent to the remaining one which we call a center vertex. We prepare N variables, $x_{1,0}, x_{1,1},\cdots,x_{1,N-1}$ to hold the number of the center vertex, and $\log D$ (where D is the degree of the center vertex) variables $y_{i,0}, y_{i,1},\cdots,y_{i,\log D-1}$ to specify the other $k-1$ vertices $(1 \le i \le k-1)$. The idea is that if some value, (say, j) is assigned to $x_{1,N-1}, x_{1,N-2},\cdots,x_{1,0}$, and if some value (say, $l, 0 \le l \le d(v_j)-1$) is assigned to $y_{i,\log D-1}, y_{i,\log D-2},\cdots,y_{i,0}$, then we

consider that the vertex which is adjacent to the vertex v_j by the lth edge out of the $d(v_j)$ ones is a member of the k-clique. To save much, we wish the degree of the center vertex to be small. The worst case is that the vertices of k largest degrees become a clique. Even so, we can select the center vertex as the vertex of the smallest degree among them.

Theorem 7. *There is a reduction from k-Clique whose SV complexity is* $\widetilde{O}(\log n + (k-1)\log D)$*, where D is the kth largest degree of the graph. (The polynomial-time reduction algorithm is omitted.)*

5.3. $\widetilde{O}(\frac{k}{3}\log T)$ Reduction

We can show another reduction (kClique-III). Again for simplicity, we explain the case of $k = 3l$ (l is a positive integer). We consider k-clique as the l collection of K_3 (triangle) subgraphs. We have to find all the possible different K_3's in the graph G, and give the number to each K_3 from 0 to $T-1$. Now we will introduce two definitions. For two different K_3's, (say, $K1$ and $K2$) if $K1$ and $K2$ contain no common vertex, we say these two K_3's are *independent*. If each of the three vertices of $K1$ is adjacent to each of $K2$ ($K1$ and $K2$ are connected by a complete bipartite subgraph), then we say these two $K_3's$ are *adjacent*. Instead of selecting k vertices as k-clique, we will select l K_3's under the condition that any two pair of K_3's are independent and adjacent. Since we need $\log T$ variables to hold each K_3 which constitute a k-clique, we need $\frac{k}{3}\log T$ variables. Thus the SV complexity of this heuristic is $\widetilde{O}(\frac{k}{3}\log T)$. The polynomial-time reduction algorithm is omitted.

Theorem 8. *There is a reduction from k-Clique whose SV complexity is* $\widetilde{O}(\frac{k}{3}\log T)$*, where T is the number of different* K_3 *in the graph.*

Remark 3. *If the average degree of G is D, then we can roughly consider that the probability that an edge exists between two vertices is* D/n*. Then the probability that any three vertices constitute* K_3 *is* $(D/n)^3$*. Since the number of possible* K_3*'s in G is* ${}_nC_3 = n^3/6$*, the expected number of* K_3*'s is* $D^3/6$*. Since* $T \leq D^3/6$*, the complexity of Theorem 8 is at most* $\widetilde{O}(k\log D)$*, which does not differ too much from the complexity of Theorem 7.*

Remark 4. *If we consider each* K_3 *as a vertex and put an edge between* K_3*'s that are independent and adjacent, then we equivalently obtain l-Clique of such as transformed graph. The heuristic of Sec.5.2 can be applied to this l-Clique problem.*

REFERENCES

[B73] C. Berge, "Graphs and hypergraphs," North-Holland, 1973.

[BCLS87] T. N. Bui, S. Chaudhuri, F. T. Leighton, and M. Sisper, "Graph bisection algorithms with good average case behavior," *Combinatorica* 7 (2),pp. 171-191, 1987.

[DCGL92] S. B-David, B. Chor, O. Goldreich, and M. Luby, "On the theory of average case complexity," *Journal of Computer and System Sciences 44*, pp. 193-219, 1992.

[DF89] M. E. Dyer and A. M. Fieze, "The solution of some random NP-hard problems in polynomial expected time," *J. of Algorithms* 10, pp. 451-489, 1989.

[J93] D. Johnson, "History of the challenge and introduction to Clique and Coloring," *DIMACS Challenge II Workshop*, Rutgers, 1993.

[KP92] E. Koutsoupias and C. Papadimitrious, "On the greedy algorithm for satisfiability," *Information Processing Letters 43*, pp. 53-55, 1992.

[L86] L. A. Levin, "Average case complete problems," *SIAM J. Comput. 15*, pp. 285-286, 1986.

[MI92] S. Miyazaki and K. Iwama, "Local search algorithms for the CNF satisfiability problem," *In* Proc. Kyushu Joint Convention, EERSJ, 1992 (in Japanese).

[MS92] J. Makowsky and A. Sharell, "On average case complexity of SAT for symmetric distributions," *Technical Report* 739, Israel Institute of Technology, Department of Computer Science, Haifa, Israel, 1992.

[SK93] B. Selman and H. T. Kautz, "Local search strategies for satisfiability testing," *DIMACS Challenge II Workshop*, Rutgers, 1993.

[SLM92] B. Selman and H. Levesque and D. Mitchell, "A new method for solving hard satisfiability problems," *In* Proc. Tenth National Conference on Artificial Intelligence, pp. 440-446, 1992.