ON THE COMPLEXITY OF MIN-MAX OPTIMIZATION PROBLEMS AND THEIR APPROXIMATION*

KER-I KO and CHIH-LONG LIN
Department of Computer Science, State University of New York at Stony Brook, Stony Brook, NY 11794.

Abstract. The computational complexity of optimization problems of the min-max form is naturally characterized by Π_2^P, the second level of the polynomial-time hierarchy. We present a number of optimization problems of this form and show that they are complete for the class Π_2^P. We also show that the constant-factor approximation versions of some of these optimization problems are also complete for Π_2^P.

1. Introduction

Consider an optimization problem of the following form:

MAX-A: for a given input x, find $\max_y\{|y| : (x, y) \in A\}$,

where A is a polynomial-time computable set such that $(x, y) \in A$ only if $|y| \leq p(|x|)$ for some polynomial p (we say A is *polynomially related*). For instance, if $A = \{(G, Q)$: $G = (V, E)$ is a graph and $Q \subseteq V$ is a clique in $G\}$, then MAX-A is the well-known maximum clique problem.[1] It is immediate that the decision version of MAX-A, i.e., the problem of determining whether $\max_y\{|y| : (x, y) \in A\}$ is greater than or equal to a given constant K, is in *NP*. In the past twenty years, a great number of optimization problems of this type have been shown to be *NP*-complete [3].[2]

Assume that $A \in P$, A is polynomially related, and that we are given m input instances $x_1, \ldots, x_m$ and are asked to find

$$\min_{1 \leq i \leq m} \max_y \{|y| : (x_i, y) \in A\}.$$

Then, (the decision version of) this problem is still in *NP*, if the instances $x_1, \ldots, x_m$ are given as input explicitly. However, if m is exponentially large relative to $|x|$ and the instances $x_1, \ldots, x_m$ have a succinct representation then the complexity of the problem may be higher than *NP*. For instance, consider the following problem MINMAX-CLIQUE: The input to the problem MINMAX-CLIQUE is a graph $G = (V, E)$ with its vertices V partitioned into subsets $V_{i,j}$, $1 \leq i \leq I$, $1 \leq j \leq J$. For any function $t : \{1, \ldots, I\} \to \{1, \ldots, J\}$, we let G_t denote the induced subgraph of G on the vertex set $V_t = \bigcup_{i=1}^{I} V_{i,t(i)}$.

* Research supported in part by NSF grant CCR 9121472.

[1] A subsete $Q \subseteq V$ is a clique of a graph $G = (V, E)$ if $\{u, v\} \in E$ for all $u, v \in Q$.

[2] Strictly speaking, their decision versions are shown to be *NP*-complete. In the rest of the paper, we will however use the term *NP*-complete for both the decision and the optimization versions of the problems.

MINMAX-CLIQUE: given a graph G with the substructures described above, find $f_{\text{CLIQUE}}(G) = \min_t \max_Q\{|Q| : Q \subseteq V \text{ is a clique in } G_t\}$.

Intuitively, the input G represents a network with I components, with each component V_i having J subcomponents $V_{i,1}, \ldots, V_{i,J}$. At any time t, only one subcomponent $V_{i,t(i)}$ of each V_i is *active*, and we are interested in the maximum clique size of G for all possible *active subgraphs* G_t of G. For people who are familiar with the *NP*-completeness theory, it is easy to see that the problem MINMAX-CLIQUE is in Π_2^P, the second level of the polynomial-time hierarchy; i.e., the decision problem of determining whether $f_{\text{CLIQUE}}(G) \leq K$, for a given constant K, is solvable by a polynomial-time nondeterministic machine with the help of an *NP*-complete set as the oracle. Therefore, it is probably not in *NP*. Indeed, we will show in Theorem 10 that this problem is complete for Π_2^P.

In general, if an input instance x contains an exponential number of subinstances $(x_1, \ldots, x_m)$ (called a *parameterized input*), then the problem of the form

MINMAX-A: for a given parametrized input x, find $f_{\text{MINMAX-A}}(x) = \min_{1 \leq t \leq m} \max_y\{|y| : (x_t, y) \in A\}$,

is a natural generalization of the problem MAX-A and its complexity is in Π_2^P. In this paper, we present a number of optimization problems of this type and show that they are complete for Π_2^P, and hence are not solvable in deterministic polynomial time even with the help of an *NP*-complete set as the oracle, assuming that the polynomial-time hierarchy does not collapse to $\Delta_2^P = P^{NP}$. These problems include the generalized versions of the maximum clique problem, the maximum 3-dimensional matching problem, the dynamic Hamiltonian circuit problem and the problem of computing the generalized Ramsey numbers.

We remark that although numerous optimization problems have been known to be *NP*-complete, there are relatively fewer natural problems known to be complete for Π_2^P (or, for Σ_2^P, the class of complements of sets in Π_2^P) (cf. [8, 12, 13, 14]). Our results here demonstrate a number of new Π_2^P-complete problems. We hope it could be a basis from which more Π_2^P-complete problems can be identified.

In the recent celebrated result of the *PCP* characterization of *NP*, Arora et al [1] showed that the constant-factor approximation versions of many optimization problems of the form MAX-A, including MAX-CLIQUE, are also *NP*-complete. It implies that if $P \neq NP$, then there is a constant $\epsilon > 0$ such that no polynomial-time algorithm can find for each input x a solution y of size $|y| \geq (1 - \epsilon)|y^*|$ such that $(x, y) \in A$, where y^* is an optimum solution for x. Through a nontrivial generalization, the *PCP* characterization of *NP* has been successfully extended to Π_2^P [2, 5, 6]. We apply this characterization to show that some of the problems of the form MINMAX-A also have similar nonapproximability property. That is, if $\Pi_2^P \neq \Delta_2^P$, then there exists a constant $\epsilon > 0$ such that no polynomial-time oracle algorithm using an *NP*-complete set as the oracle can compute, for each x, a value k such that $k^*/(1 + \epsilon) \leq k \leq (1 + \epsilon)k^*$, where $k^* = f_{\text{MINMAX-A}}(x)$. These problems include the min-max versions of the maximum clique problem, the maximum 3-dimensional matching problem and the longest circuit problem.

2. Definitions

In this section, we review the notion of Π_2^P-completeness, and present the definition of the optimization problems. In the following, we assume that the reader is familiar with the complexity classes P, NP and the notion of NP-completeness. For the formal definitions and examples, the reader is referred to any standard text, for instance, [3].

For any string x in $\{0,1\}^*$, we denote by $|x|$ the length of x. Let $\langle x, y\rangle$ be any *pairing* function, i.e., a one-to-one mapping from strings x and y to a single string in polynomial time. A well-known characterization of the class NP is as follows: $A \in NP$ if and only if there exists a set $B \in P$ such that for all $x \in \{0,1\}^*$,

$$x \in A \iff (\exists y, |y| \leq p(|x|))\langle x, y\rangle \in B,$$

where $p(n)$ is some polynomial depending only on A. The complexity class Π_2^P is a natural extension of the class NP: $A \in \Pi_2^P$ if and only if there exists a set $B \in P$ such that

$$x \in A \iff (\forall y, |y| \leq p(|x|))(\exists z, |z| \leq p(|x|))\langle x, \langle y, z\rangle\rangle \in B.$$

It is obvious that $NP \subseteq \Pi_2^P$. Between the complexity classes NP and Π_2^P lies the complexity class Δ_2^P (or, P^{NP}) that consists of all problems that are solvable in polynomial time with the help of an NP-complete problem as the oracle. Whether $NP = \Delta_2^P$ and/or $\Delta_2^P = \Pi_2^P$ are major open questions in complexity theory.

A decision problem A is Π_2^P*-complete*, if $A \in \Pi_2^P$, and for every $A' \in \Pi_2^P$, there is a polynomial-time computable function f such that for each $x \in \{0,1\}^*$, $x \in A' \iff f(x) \in A$ (f is called a *reduction* from A' to A). There are a few natural problems known to be complete for Π_2^P. A standard Π_2^P-complete problem that will be used in our proofs is the following generalization of the famous NP-complete problem SAT. Suppose that F is a 3-CNF boolean formula. We write $F(X, Y)$ to emphasize that its variables are partitioned into two sets X and Y. For a 3-CNF boolean formula $F(X, Y)$, and for any truth assignments $\tau_1 : X \to \{0,1\}$ and $\tau_2 : Y \to \{0,1\}$, we write $F(\tau_1, \tau_2)$ to denote the formula F with its variables taking the truth values defined by τ_1 and τ_2. We also write $tc(F(\tau_1, \tau_2))$ to denote the number of clauses of F that are true to the truth assignments τ_1 and τ_2.

> SAT$_2$: for a given 3-CNF boolean formula $F(X, Y)$, determine whether it is true that for all truth assignments $\tau_1 : X \to \{0,1\}$, there is a truth assignment $\tau_2 : Y \to \{0,1\}$ such that $F(\tau_1, \tau_2) = 1$.[3]

Proposition 1 SAT$_2$ *is* Π_2^P*-complete.*

The problem SAT$_2$ may be viewed as (a subproblem of) the decision version of the following optimization problem:

> MINMAX-SAT: for a given 3-CNF boolean formula $F(X, Y)$, find $f_{\text{SAT}}(F) = \min_{\tau_1 : X \to \{0,1\}} \max_{\tau_2 : Y \to \{0,1\}} tc(F(\tau_1, \tau_2))$.

[3] Throughout the paper, we identify 1 with *true* and 0 with *false*.

In the following, we introduce some new optimization problems of the min-max form. For each optimization problem, we also list its corresponding decision version. First, we consider the problem MINMAX-SAT in the restricted form.

MINMAX-SAT-B: for a given 3-CNF boolean formula $F(X, Y)$ in which each variable occurs in at most b clauses where b is a constant independent of the size of F, find $f_{\mathrm{SAT}}(F)$. (Decision version: for an additional input $K > 0$, is $f_{\mathrm{SAT}}(F) \geq K$?)

In addition to the problem MINMAX-CLIQUE defined in Section 1, we introduce a few more min-max optimization problems that are the generalizations of some famous *NP*-complete optimization problems based on the idea of parameterized inputs. Recall that for a graph $G = (V, E)$ with its vertex set V partitioned into subsets $V_{i,j}$, $1 \leq i \leq I$, $1 \leq j \leq J$, and for a function $t : \{1, \ldots, I\} \to \{1, \ldots, J\}$, we let $V_t = \bigcup_{i=1}^{I} V_{i,t(i)}$ and let G_t be the induced subgraph of G on the vertex set V_t. The following generalized vertex cover problem is a dual problem of MINMAX-CLIQUE. For a graph $G = (V, E)$, we say that a subset $V' \subseteq V$ is a *vertex cover* if $V' \cap \{u, v\} \neq \emptyset$ for all edges $\{u, v\} \in E$.

MAXMIN-VC: given a graph G with its vertex set V partitioned into subsets $\{V_{i,j}\}_{1 \leq i \leq I, 1 \leq j \leq J}$, find $f_{\mathrm{VC}}(G) = \max_t \min_{V'}\{|V'| : V' \subseteq V_t$ is a vertex cover of $G_t\}$. (Decision version: Is $f_{\mathrm{VC}}(G) \leq K$?)

The following problem is the generalization of the Hamiltonian circuit problem. For a graph $G = (V, E)$ and a subset $V' \subseteq V$, we say G *has a circuit on* V' if there is a cycle of G going through each vertex of V' exactly once. We say G is *Hamiltonian* if G has a circuit on V.

MINMAX-CIRCUIT: given a graph G with its vertex set V partitioned into subsets $\{V_{i,j}\}_{1 \leq i \leq I, 1 \leq j \leq J}$, find $f_{\mathrm{CIRCUIT}}(G) = \min_t \max_{V'}\{|V'| : V' \subseteq V_t, G_t$ has a circiut on $V'\}$. (Decision version: Is $f_{\mathrm{CIRCUIT}}(G) \geq K$?)

The next problem is the generalization of the maximum 3-dimensional matching problem. Let W be a finite set and S be a collection of 3-element subsets of W. Let W' be a subset of W. A subset $S' \subseteq S$ is called a (3-*dimensional*) *matching in* W' if all sets $s \in S'$ are mutually disjoint, and are contained in W'. In the following, if $W = \bigcup_{i=1}^{I} \bigcup_{j=1}^{J} W_{i,j}$, and t is a function from $\{1, \ldots, I\}$ to $\{1, \ldots, J\}$, we write $W(t)$ to denote the set $\bigcup_{i=1}^{I} W_{i,t(i)}$.

MINMAX-3DM: given mutually disjoint finite sets $W_{i,j}$, $1 \leq i \leq I$, $1 \leq j \leq J$, and a set S of 3-element subsets of $W = \bigcup_{i=1}^{I} \bigcup_{j=1}^{J} W_{i,j}$, find $f_{\mathrm{3DM}}(W, S) = \min_t \max_{S'}\{|S'| : S' \subseteq S, S'$ is a matching in $W(t)\}$, where t ranges over all functions from $\{1, \ldots, I\}$ to $\{1, \ldots, J\}$. (Decision version: Is $f_{\mathrm{3DM}}(W, S) \geq K$?)

In addition to the above problems based on the idea of parameterized inputs, we consider several natural problems in Π_2^P. First, we consider the problem of computing the Ramsey number. For any graph $G = (V, E)$, a function $c : E \to \{0, 1\}$ is called a *coloring* of G (with two colors). For any complete graph G with a two-color coloring c, a set $Q \subseteq V$ is a *monochromatic clique* if all edges between vertices in Q are of the same color. Ramsey theorem states that for any positive integer K, there exists an integer $n = R_K$ such that for all two-colored complete graph G of

size n, there is a monochromatic clique Q of size K. To study the complexity of computing the Ramsey function mapping K to the minimum R_K, we consider the following generalized version. In the following, we say a function $c : E \to \{0, 1, *\}$ is a *partial coloring* of a graph $G = (V, E)$ ($c(e) = *$ means the edge e is not colored yet). A coloring $c' : E \to \{0, 1\}$ is a *restriction* of a partial coloring c, denoted by $c' \preceq c$, if $c'(e) = c(e)$ whenever $c(e) \neq *$.

GENERALIZED RAMSEY NUMBER(GRN): given a complete graph $G = (V, E)$ with a partial coloring c, find $f_{\mathrm{GRN}}(G, c) = \min_{c' \preceq c} \max_Q \{|Q| : Q$ is a monochromatic clique under $c'\}$. (Decision version: Is $f_{\mathrm{GRN}}(G, c) \geq K$?)

Notice that the Ramsey number R_K can be found by a binary search for the graph G of the minimum size that has $f_{\mathrm{GRN}}(G, c_0) \geq K$ with respect to the empty coloring c_0 (i.e., $c_0(e) = *$ for all edges e).

The next two problems are the variations of the Hamiltonian circuit problem. The first problem is to find, from a given digraph and a subset of *alterable* edges, the length of the longest circuit in any alteration of those edges. Let $G = (V, E)$ be a digraph and D a subset of E. We let G_D denote the subgraph of G with vertex set V and edge set $(E - D) \cup \mathrm{inv}(D)$, where $\mathrm{inv}(D) = \{(s, t) : (t, s) \in D\}$.

LONGEST DIRECTED CIRCUIT (LDC): given a digraph $G = (V, E)$ and a subset E' of E, find $f_{\mathrm{LDC}}(G, E') = \min_{D \subseteq E'} \max_{V'} \{|V'| : V' \subseteq V, G_D$ has a circuit on $V'\}$. (Decision version: Is $f_{\mathrm{LDC}}(G, E') \geq K$?)

The next problem is similar to the above problem, but is about the longest circuits in undirected graphs. For simplicity, we formulate it as a special case of its decision version.

DYNAMIC HAMILTONIAN CIRCUIT (DHC): given a graph $G = (V, E)$, and a subset B of E, determine whether $G_D = (V, E - D)$ is Hamiltonian for all subsets D of B with$|D| \leq |B|/2$.

3. Π_2^P-Completeness Results

All the problems defined in Section 2 can be easily seen belonging to Π_2^P. In this section, we show that MINMAX-CIRCUIT, GRN and DHC are actually Π_2^P-complete. The problems MINMAX-CLIQUE and MINMAX-3DM will be shown to be Π_2^P-complete in Section 5 together with the stronger results that their contant-factor approximation versions are also Π_2^P-complete. The Π_2^P-completeness of MAXMIN-VC is a corollary of that of MINMAX-CLIQUE. The proofs for the Π_2^P-completeness of the problems MINMAX-SAT-B and LDC are much more involved; we prove them in a separate paper [7].

Theorem 2 MINMAX-CIRCUIT *is complete for* Π_2^P.

Proof. We construct a reduction from SAT_2 to (the decision version of) MINMAX-CIRCUIT. The construction is a modification of the reduction from SAT to the Hamiltonian circuit problem. Let F be a 3-CNF boolean formula over variables $X = \{x_1, \ldots, x_r\}$ and $Y = \{y_1, \ldots, y_s\}$. Assume that $F = C_1 \wedge C_2 \wedge \cdots \wedge C_n$, where

each C_j is the OR of three literals. We will define a graph G over $18n + 4r + 2s$ vertices.

For each clause C_j, we define a subgraph H_j of 18 vertices as shown in Figure 1. This subgraph H_j will be connected to other parts of the graph G only through the vertices labeled $\alpha_k[j]$ and $\beta_k[j]$, $k = 1, 2, 3$. Thus, it has the following property: if a Hamiltonian circuit of G enters H_j through $\alpha_k[j]$ for some $k = 1, 2, 3$, then it must exit at $\beta_k[j]$, and visit either one or two or all three rows of H_j.

In addition to subgraphs H_j, we have some more vertices: For each variable x_i in X, we define four vertices: $u_{i,0}, u_{i,1}, \bar{u}_{i,0}, \bar{u}_{i,1}$. For each variable y_i in Y, we define two vertices: $v_{i,0}, v_{i,1}$.

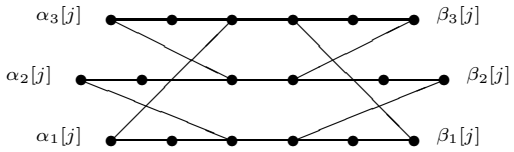

Fig. 1. The subgraph H_j for a clause C_j.

We define the edges between these components as follows.

(1) For each i, $1 \le i < r$, we define edges $\{u_{i,1}, u_{i+1,0}\}$, $\{u_{i,1}, \bar{u}_{i+1,0}\}$, $\{\bar{u}_{i,1}, u_{i+1,0}\}$, $\{\bar{u}_{i,1}, \bar{u}_{i+1,0}\}$. For $i = r$, we have two edges $\{u_{r,1}, v_{1,0}\}$, $\{\bar{u}_{r,1}, v_{1,0}\}$.

(2) For each i, $1 \le i < s$, we define an edge $\{v_{i,1}, v_{i+1,0}\}$. For $i = s$, we have two edges $\{v_{s,1}, u_{1,0}\}$, $\{v_{s,1}, \bar{u}_{1,0}\}$.

(3) For each literal z and for $k = 0, 1$, let

$$w(z)_k = \begin{cases} u_{i,k} & \text{if } z = x_i, \\ \bar{u}_{i,k} & \text{if } z = \bar{x}_i, \\ v_{i,k} & \text{if } z = y_i \text{ or } \bar{y}_i. \end{cases}$$

Then we define edges to form a path from $w(z)_0$ to $w(z)_1$: assume that z occurs as the k_1th, k_2th, ..., k_mth literal in clauses $C_{j_1}, C_{j_2}, \ldots, C_{j_m}$, respectively, with $j_1 < j_2 < \cdots < j_m$. Then we add edges $\{w(z)_0, \alpha_{k_1}[j_1]\}$, $\{\beta_{k_m}[j_m], w(z)_1\}$, and for each $\ell = 1, \ldots, m-1$, $\{\beta_{k_\ell}[j_\ell], \alpha_{k_{\ell+1}}[j_{\ell+1}]\}$. (Note that for each pair $(u_{i,0}, u_{i,1})$ or $(\bar{u}_{i,0}, \bar{u}_{i,1})$, there is a path between them, and for each pair $(v_{i,0}, v_{i,1})$ there are two paths between them, corresponding to the occurrences of two literals y_i and $\bar{y}_i$.)

The above completes the definition of all edges. To complete the reduction, we let $I = r + 1$, $J = 2$, and for each $1 \le i \le r$, $V_{i,0} = \{\bar{u}_{i,0}, \bar{u}_{i,1}\}$ and $V_{i,1} = \{u_{i,0}, u_{i,1}\}$, and all other vertices are in $V_{r+1,0} = V_{r+1,1}$. (For convenience, we define $V_{i,0}$ and $V_{i,1}$ instead of $V_{i,1}$ and $V_{i,2}$.) Finally, we let $K = 18n + 2r + 2s$, which is equal to the size of $|V_t|$ for all functions $t : \{1, \ldots, I\} \to \{0, 1\}$.

The correctness of this reduction is very easy to see. We only give a short sketch here. First, assume that for each truth assignment τ_1 on X, there is a truth assignment τ_2 on Y satisfying all clauses C_j in F. Let $t : \{1, \ldots, r+1\} \to \{0, 1\}$ be any function. Then, t defines a truth assignment $\tau_1(x_i) = t(i)$, and all vertices in $V_{i,t(i)}$ corresponds to "true" literals under τ_1. From this τ_1, there is a truth

assignment τ_2 on Y that satisfies all clauses. Now, for each "true" literal z under τ_1 and τ_2, we have a path from $w(z)_0$ to $w(z)_1$ in G_t. Connecting these paths together forms a Hamiltonian circuit for G_t, since each subgraph H_j is visited by at least one such paths.

Conversely, assume that for any $t : \{1, \ldots, I\} \to \{0, 1\}$, there is a Hamiltonian circuit Π_t in G_t. Then, by the basic property of subgraphs H_j, this circuit Π_t defines, for each pair of nodes $(w(z)_0, w(z)_1)$ in V_t, a path from $w(z)_0$ to $w(z)_1$. That is, for each τ_1 on X such that $\tau_1(x_i) = t(i)$, we can define a truth assignment τ_2 on Y by $\tau_2(y_i) = 1$ (or, $\tau_2(y_i) = 0$) if the path of Π_t from $v_{i,0}$ to $v_{i,1}$ visits nodes corresponding to the literal y_i (or, respectively, $\bar{y}_i$). Since each subgraph H_j is visited by at least one of such paths, the assignments τ_1 and τ_2 together must satisfy each clause C_j. □

Theorem 3 GRN *is complete for* Π_2^P.

Proof. We construct a reduction from SAT_2 to GRN. Let F be a 3-CNF formula over variables $X = \{x_1, \ldots, x_r\}$ and $Y = \{y_1, \ldots, y_s\}$. Assume that $F = C_1 \wedge C_2 \wedge \cdots \wedge C_n$, where each C_i is the OR of three literals. We further assume that $r \geq 2$ and $n \geq 3$. Let $K = 2r+n$. The graph G has $N = 6r+4n-4$ vertices. We divide them into three groups: $V_X = \{x_{i,j}, \bar{x}_{i,j} : 1 \leq i \leq r, 1 \leq j \leq 2\}$, $V_C = \{c_{i,j} : 1 \leq i \leq n, 1 \leq j \leq 3\}$ and $V_R = \{r_i : 1 \leq i \leq 2r + n - 4\}$. The partial coloring c on the edges of G is defined as follows (we use colors *blue* and *red* instead of 0 and 1):

(1) The edges among $x_{i,1}$, $x_{i,2}$, $\bar{x}_{i,1}$ and $\bar{x}_{i,2}$, for each i, $1 \leq i \leq r$, are colored by red, except that the edges $e_i = \{x_{i,1}, x_{i,2}\}$ and $\bar{e}_i = \{\bar{x}_{i,1}, \bar{x}_{i,2}\}$ are not colored (i.e., $c(e_i) = c(\bar{e}_i) = *$).

(2) All other edges between two vertices in V_X are colored by blue; i.e., $c(\{x_{i,j}, x_{i',j'}\}) = c(\{x_{i,j}, \bar{x}_{i',j'}\}) =$ blue if $i \neq i'$.

(3) All edges among vertices in V_R are colored by red.

(4) For each i, $1 \leq i \leq k$, the three edges among $c_{i,1}, c_{i,2}$ and $c_{i,3}$ are colored by red.

(5) The edge between two vertices $c_{i,j}$ and $c_{i',j'}$, where $i \neq i'$, is colored by red if the jth literal of C_i and the j'th literal of $C_{i'}$ are complementary (i.e., one is x_q and the other is $\bar{x}_q$, or one is y_q and the other is $\bar{y}_q$ for some q). Otherwise, it is colored by blue.

(6) The edge between any vertex in V_R and any vertex in V_X is colored by red, and the edge between any vertex in V_R and any vertex in V_C is colored by blue.

(7) For each vertex $c_{i,j}$ in V_C, if the jth literal of C_i is y_q or $\bar{y}_q$ for some q, then all edges between $c_{i,j}$ and any vertex in V_X are colored by blue. If the jth literal of C_i is x_q for some q, then all edges between $c_{i,j}$ and any vertex in V_X, except $\bar{x}_{q,1}$ and $\bar{x}_{q,2}$, are colored by blue, and $c(\{c_{i,j}, \bar{x}_{q,1}\}) = c(\{c_{i,j}, \bar{x}_{q,2}\}) =$ red. The case where the jth literal of C_i is $\bar{x}_q$ for some q is symmetric; i.e., all edges between $c_{i,j}$ and any vertex in V_X, except $x_{q,1}$ and $x_{q,2}$, are colored by blue, and $c(\{c_{i,j}, x_{q,1}\}) = c(\{c_{i,j}, x_{q,2}\}) =$ red.

The above completes the construction of the graph G and its partial coloring c. Notice that the partial coloring c has $c(e) \neq *$ for all edges e except e_i and $\bar{e}_i$, for $1 \leq i \leq r$. Now we prove that this construction is correct. First assume that for

each assignment $\tau_1 : X \to \{0,1\}$, there is an assignment $\tau_2 : Y \to \{0,1\}$ such that $F(\tau_1, \tau_2) = 1$. We verify that for any two-coloring restriction c' of c, there must be a size-K monochromatic clique Q.

We note that if $c'(e_i) = c'(\bar{e}_i) =$ red for some $i \le r$, then the vertices $x_{i,1}$, $x_{i,2}$, $\bar{x}_{i,1}$, $\bar{x}_{i,2}$, together with vertices in V_R, form a red clique of size $|V_R| + 4 = K$. Therefore, we may assume that for each i, $1 \le i \le r$, at least one of $c'(e_i)$ and $c'(\bar{e}_i)$ is blue. Now we define an assignment τ_1 on X by $\tau_1(x_i) = 1$ if and only if $c'(e_i) =$ blue. For this assignment τ_1, there is an assignment τ_2 on Y such that each clause C_i has a true literal. For each i, $1 \le i \le k$, let j_i be the least j, $1 \le j \le 3$, such that the jth literal of C_i is true to τ_1 and τ_2. Let $Q_C = \{c_{i,j_i} : 1 \le i \le n\}$ and $Q_X = \{x_{i,j} : c'(e_i) = \text{blue}, 1 \le j \le 2\} \cup \{\bar{x}_{i,j} : c'(e_i) = \text{red}, 1 \le j \le 2\}$. Let $Q = Q_C \cup Q_X$. It is clear that Q is of size $2r + n$. Furthermore, Q is a blue clique: (i) every two vertices in Q_C are connected by a blue edge because they both have value true under τ_1 and τ_2 and so are not complementary; (ii) every two vertices in Q_X are connected by a blue edge by the definition of Q_X, and (iii) if a vertex c_{i,j_i} in Q_C corresponds to a literal x_q, then $\tau_1(x_q) = 1$ and so $\bar{x}_{q,1}, \bar{x}_{q,2} \notin Q_X$ and hence all the edges between c_{i,j_i} and each of $x_{i',j'}$ or $\bar{x}_{i',j'} \in Q_X$ are colored blue.

Conversely, assume that there exists an assignment τ_1 on X such that for all assignments τ_2 on Y, $F(\tau_1, \tau_2) = 0$. Then, consider the following coloring c' on edges e_i and $\bar{e}_i$: $c'(e_i) =$ blue and $c'(\bar{e}_i) =$ red if $\tau_1(x_i) = 1$, and $c'(e_i) =$ red and $c'(\bar{e}_i) =$ blue if $\tau_1(x_i) = 0$. By the definition of c', the largest red clique in V_X is of size 3. Also, the largest red clique in V_C is of size 3, since every edge connecting two noncomplementary literals in two different clauses is colored by blue. Thus, the largest red clique containing V_R is of size $K-1$, and the largest red clique containing at least one vertex of V_C is of size $\le 6 < K$.

Next, assume by way of contradiction that there is a blue clique Q of G of size K. From our coloring, it is clear that, for each i, $1 \le i \le r$, Q contains exactly two vertices in $\{x_{i,1}, x_{i,2}, \bar{x}_{i,1}, \bar{x}_{i,2}\}$, and for each i, $1 \le i \le n$, Q contains exactly one c_{i,j_i}, for some $1 \le j_i \le 3$. Define $\tau_2 : Y \to \{0,1\}$ by $\tau_2(y_q) = 1$ if and only if the j_ith literal of C_i is y_q for some i, $1 \le i \le k$. Then, there is a clause C_i such that C_i is not satisfied by τ_1 and τ_2. In particular, the j_ith literal of C_i is false to τ_1 and τ_2.

Case 1. The j_ith literal of C_i is x_q for some q. Then, $\tau_1(x_q) = 0$, and so $c'(e_q) =$ red, and the edges between c_{i,j_i} and each of $\bar{x}_{q,1}$ and $\bar{x}_{q,2}$ are red. This contradicts the above observation that Q contains two vertices in $\{x_{q,1}, x_{q,2}, \bar{x}_{q,1}, \bar{x}_{q,2}\}$

Case 2. The j_ith literal of C_i is $\bar{x}_q$ for some q. This is symmetric to Case 1.

Case 3. The j_ith literal of C_i is y_q for some q. This is not possible, because by the definition of τ_2, $\tau_2(y_q) = 1$, but by the property that C_i is not satisfied by τ_1 and τ_2, $\tau_2(y_q) = 0$.

Case 4. The j_ith literal of C_i is $\bar{y}_q$ for some q. Then, $\tau_2(y_q) = 1$, and hence, by the definition of τ_2, there must be another $i' \le n$, $i' \ne i$, such that $c_{i',j_{i'}}$ is in Q and the $j_{i'}$th literal of $C_{i'}$ is y_q. So, the edge between c_{i,j_i} and $c_{i',j_{i'}}$ is colored by red. This is again a contradiction.

The above case analysis shows that there is no blue clique in G of size K either. So the theorem is proven. □

Theorem 4 DHC *is* Π_2^{P}*-complete.*

Proof. We reduce SAT_2 to DHC. Let $F = C_1 \wedge C_2 \wedge \cdots \wedge C_n$ be a boolean formula over variables in $X = \{x_1, \ldots, x_r\}$ and $Y = \{y_1, \ldots, y_s\}$ where C_i's are three-literal clauses. We construct a graph G from F. A basic component of the graph G is a NOT device as shown in Figure 2(a). (It was first introduced in [9], and was called an exclusive-OR device in [11]). Schematically, the two horizontal line paths are represented by two *broadened* line segments, one designated as input and the other output to the device, and the four vertical paths are "condensed" into one *arrow*, running from input to output (see Figure 2(c)). As shown in [9], there are only two ways for a Hamiltonian circuit to traverse such a device, one of them indicated in Figure 2(a) by thicker line segments. For convenience, if a NOT device is traversed as shown in Figure 2(a), we say that the NOT device has input *true*.

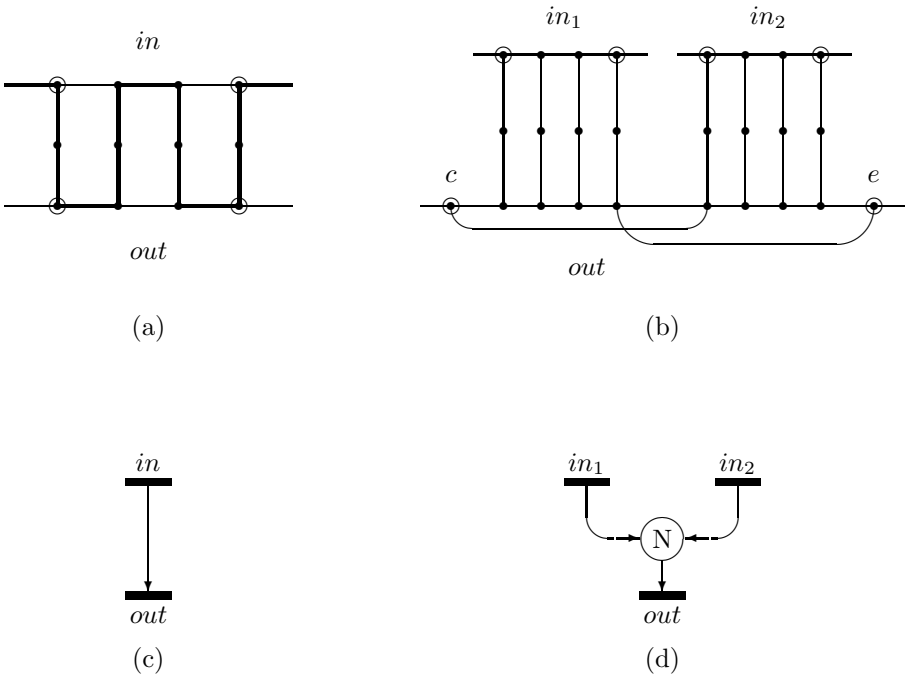

Fig. 2. The NOT and NAND devices.

Using two NOT devices, we also have the NAND device as shown in Figure 2(b) and 2(d). Here, in order for a Hamiltonian circuit to traverse from c to e or vice versa, *at least one* of the two input NOT devices have to be false. When using these devices, we require that connections to other parts of the graph G can only go through *circled* vertices.

The graph G we are going to construct consists of a set of interconnected subgraphs:

(1) For each $x_i \in X$, we have a *variable subgraph* $G_x(i)$ as shown in Figure 3(a), and for each $y_j \in Y$, a $G_y(j)$ as in Figure 3(b). The number of NOT devices

used in each variable subgraph will be defined in (3) below. In addition, we have a *component subgraph* G_w, which is a concatenation of $n+r$ NOT devices as shown in Figure 3(c). Note that each variable or component subgraph has some extra labeled vertices. They are not part of any NOT devices, except that c and e of $G_x(i)$ are precisely those in the NAND device shown in figure 2(b).

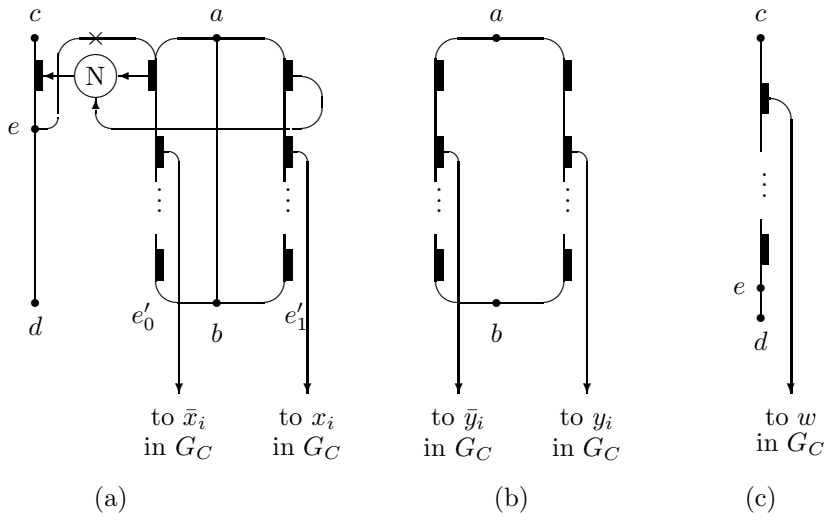

Fig. 3. The variable subgraphs: (a) $G_x(i)$, (b) $G_y(i)$ and (c) G_w.

(2) For each clause C_k, $1 \le k \le n$, we have a *clause subgraph* $G_C(k)$as shown in Figure 4. In addition, we have r extra clause subgraphs $G_C(n+1), \ldots, G_C(n+r)$, each of which is a *triangle* version of Figure 4. Note that if the four *conner vertices* are further connected as a clique, then any graph containing such a clause subgraph is Hamiltonian only if at least one of the four (or three) NOT devices has its input true.

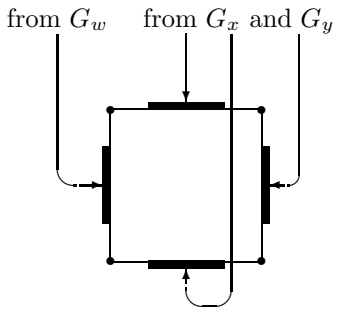

Fig. 4. The clause subgraph $G_C(i)$ that has two literals from X and one from Y.

(3) Arrows (of NOT devices) run from variable subgraphs to clause subgraphs as follows:

(a) If literal $\bar{x}_i$ (or x_i) occurs in m clauses, then there are $m+2$ NOT devices on the *negative* (or, respectively, *positive*) path, the path containing e'_0 (or, respectively, e'_1), of $G_x(i)$. The inputs of the first NOT devices on both paths serve as inputs to a NAND device as shown in Figure 3(a).

(b) The numbers of NOT devices in G_y's are defined analogously except that no NAND devices are involved.

(c) An arrow runs from the input of a NOT device in a variable subgraph to the output of a NOT device in a clause subgraph if and only if the corresponding variable, either positive or negative, occurs in that clause. For example, if $C_k = (x_i \vee \cdots)$, then there is an arrow running from the input of a NOT device on the positive path of $G_x(i)$ to the output of a NOT device in $G_C(k)$. Furthermore, for each $G_C(n+i)$, $1 \le i \le r$, we have two inputs from $G_x(i)$, one corresponding to x_i and the other $\bar{x}_i$, and another input from one of the NOT devices in G_w.

(4) Let $a_x(i)$ denote the vertex of $G_x(i)$ labeled with a in Figure 3(a). Let $b_x(i)$, $c_x(i)$, $d_x(i)$, $e_x(i)$, $a_y(j)$, $b_y(j)$, $e'_0(i)$, $e'_1(i)$, c_w, e_w and d_w be defined analogously. Then, in addition to those arrows going to G_C from G_x, G_y and G_w, these subgraphs are further connected by the following edges (see Figure 5):

$\langle b_x(i), a_x(i+1)\rangle$, $\langle c_x(i), a_x(i+1)\rangle$ and $\langle d_x(i), c_x(i+1)\rangle$ for $1 \le i \le r-1$;
$\langle b_y(j), a_y(j+1)\rangle$ for $1 \le j \le s-1$;
$\langle b_x(r), a_y(1)\rangle$ and $\langle c_x(r), a_y(1)\rangle$;
$\langle d_x(r), c_w\rangle$, $\langle c_w, a_x(1)\rangle$ and $\langle d_w, c_x(1)\rangle$;
edges that connect all the corner vertices of all clause subgraphs into a clique;
$\langle b_y(s), a_C\rangle$, $\langle b_C, e_w\rangle$, and $\langle b_C, a_x(1)\rangle$, where a_C and b_C are two distinguished corner vertices in two different clause subgraphs.

The graph G is now completed. Finally we define B to be the set of edges $e'_0(i)$ and $e'_1(i)$ in $G_x(i)$, for all $1 \le i \le r$. This finishes the construction.

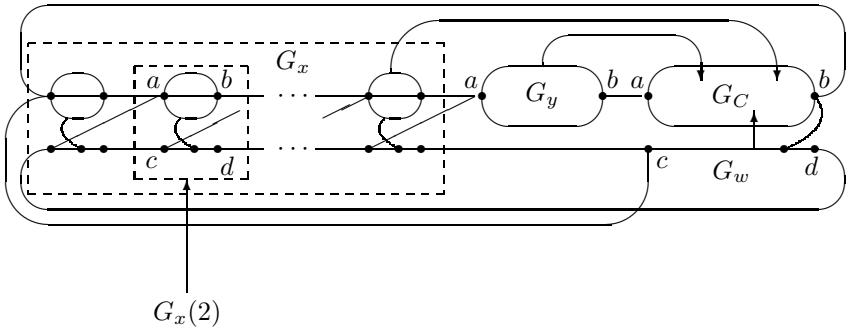

Fig. 5. The interconnection among subgraphs.

We now show that the above construction is a reduction from SAT_2 to DHC. First suppose that $F \in \text{SAT}_2$. We claim that G_D is Hamiltonian for any $D \subseteq B$ with $|D| \le |B|/2$. Consider the following two cases:

Case 1. For some i, $1 \le i \le r$, neither $e'_0(i)$ nor $e'_1(i)$ is in D. We show the existence of a Hamiltonian circuit H in this case. Let i be the smallest index affirming the case. Starting from $a_x(1)$, H visits all vertices in G_D in the following order:

(i) H first visits $a_x(1), b_x(1), a_x(2), b_x(2), \ldots, a_x(i-1)$, $b_x(i-1)$ and then $a_x(i)$.

(ii) It clockwisely visits all the NOT devices on the "loop" of $G_x(i)$, making the inputs to these NOT devices true, take the edge marked with $\times$ (see Figure 3(a)) to $e_x(i)$ and leaves at $d_x(i)$.

(iii) It then proceeds to visit $c_x(i+1)$, $d_x(i+1)$, $\ldots$, $c_x(r)$, $d_x(r)$, c_w, d_w, $c_x(1)$, $d_x(1)$, $\ldots$, $c_x(i-1)$, $d_x(i-1)$, $c_x(i)$. While going from each $c_x(k)$ to $d_x(k)$, $k \neq i$, it traverses the NAND device in between (and so sets both inputs false), and from c_w to d_w, sets all the NOT devices in between true.

(iv) It then visits $a_x(i+1), b_x(i+1), \ldots, a_x(r), b_x(r)$, and then sets all y_j to false (i.e., visits all the NOT devices on the left half of each $G_y(j)$).

(v) H finally visits G_C completely, starting at a_C and leaving via b_C, and returns to $a_x(1)$. Note that we are able to do so because all clause subgraphs contain the output of a NOT device whose input is in G_w, which had been set to true previously; further, all NOT devices in variable subgraphs not traversed before are visited here.

Case 2. Exactly one of $e'_0(i)$ and $e'_1(i)$ is in D for all i, $1 \le i \le r$. The Hamiltonian circuit H in this case is as follows. It starts from $a_x(1)$.

(i) It chooses the positive (negative) path of $G_x(i)$ if $e'_1(i)$ (respectively, $e'_0(i)$) is not in D, and visits all the NOT devices on the path.

(ii) Since the edges $e'_0(i)$ or $e'_1(i)$ not in D correspond to a truth assignment τ_1 on X (i.e., $\tau_1(x_i) = 1$ if and only if $e'_1(i)$ is not in D), there exists a truth assignment τ_2 on Y such that $F(\tau_1, \tau_2)$ is true. H sets each y_j accordingly by traversing the positive (or negative) path of $G_y(j)$ if y_j is set to true (or, respectively, false).

(iii) H then traverses G_C completely, leaving via b_C. We are able to do so because $F(\tau_1, \tau_2)$ is true and so each $G_C(j)$, $1 \le j \le n$, has at least one NOT device having the true input. Further, for each $G_C(n+i)$, $1 \le i \le r$, one of the NOT devices corresponding to x_i or $\bar{x}_i$ has its input true. All NOT devices in variable subgraphs not traversed previously are done at this stage.

(iv) It visits e_w, then visits d_w, $c_x(1)$, $d_x(1)$, $\ldots$, $c_x(r)$, and finally $d_x(r)$, traversing each NAND device between $c_x(k)$ and $e_x(k)$, since exactly one input has been set true. It leaves via c_w and finishes at $a_x(1)$.

Since $|D| \le |B|/2$, we cannot have $e'_0(i)$ and $e'_1(i)$ both in D without having $e'_0(i')$ and $e'_1(i')$ both not in D for some i'. Therefore, the above two cases are exhaustive.

Suppose on the other hand that for some truth assignment τ_1 on X, $F(\tau_1, Y)$ is not satisfiable. We let $e'_0(i) \in D$ if $t(x_i) = 1$, and $e'_1(i) \in D$ otherwise. We need to show that G_D is not Hamiltonian. Suppose, for the sake of contradiction, that G_D has a Hamiltonian circuit H. First we can exclude the possibility that H enters and leaves a NOT device (including those within an NAND device) on different sides. Thus we can virtually "ignore" those arrows as far as H is concerned. As a consequence, either all the NOT devices in G_w are set to true by H or all set to false by H. We consider these two cases separately.

Case 1. All NOT devices in G_w are set to true by H. Orientation ignored, we assume that H starts from c_w, visiting all NOT devices, and then e_w. H then has to take d_w since taking b_C will leave d_w unvisited. Next it must visit $c_x(1)$, the only choice.

To visit other parts of G_D, H eventually has to leave the bottom loop which contains all the distinguished vertices labeled with c, e or d. It cannot leave from a vertex $e_x(j)$ since this would keep $d_x(j)$ out of the reach. Therefore, it must leave at $c_x(k)$ for some $1 \leq k \leq r$. It then has to visit $e_x(k)$ later. It means that the two inputs to the NAND device between $c_x(k)$ and $e_x(k)$ must be set to true, which is impossible because only one of $e'_0(k)$ and $e'_1(k)$ is in G_D.

Case 2. All of the NOT devices in G_w are set to false by H. Orientation ignored, we assume that H starts at $a_x(1)$, visiting each variable subgraph. There are two possibilities:

(a) If during traversing $G_x(i)$, H enters the bottom loop at either $c_x(i)$ or $e_x(i)$, then it has to leave the loop before visiting $d_x(r)$; otherwise, it would have to visit c_w and then either ends its journey prematurally or visits G_w, contradictary to our assumption. A contradiction can be derived as in Case 1.

(b) H visits all variable subgraphs in a normal manner (i.e., not going to the bottom loop from $G_x(i)$'s). Then H defines a truth assignment on X and Y which is consistent with τ_1 when restricted to X. By asumption, at least one of the clause C_k of F is not satisfied, meaning that none of the NOT devices of $G_C(k)$ has true input. As indicated previously, H then cannot traverse $G_C(k)$ completely, which is a contradiction. □

4. Approximation Problems and Their Hardness

We first formalize the notion of approximating optimization problems. Garey and Johnson [3] have defined the notion of approximating optimization problems of the form MAX-A. Since our min-max optimization problems have two parameters, their definition is not suitable to our case. In the following, we define a simple notion of *approximating a function*. Let $\mathbf{Q}^+$ be the set of positive rationals and $\mathbf{R}^+$ the set of positive reals.

Definition 5 *Let $f, g : \{0,1\}^* \to \mathbf{Q}^+$ and $c : \mathbf{N} \to \mathbf{R}^+$, $c(n) > 1$ for all n, be given. We say that g* approximates *f to* within a factor of *c (c-*approximates *f in short) if for all $x \in \{0,1\}^*$, we have $f(x)/c(|x|) < g(x) < c(|x|) \cdot f(x)$. The c-*approximation problem *of f is to compute a function g that c-approximates f.*

To define the notion of completeness of c-approximation problem of a function f, we generalize the notion of reductions between decision problems. In the following, we write $\langle A, B\rangle$ to denote a pair of sets over $\{0,1\}$ with $A \cap B = \emptyset$.

Definition 6 *(a) A pair $\langle A, B\rangle$ is* polynomial-time separable *(or, simply, $\langle A, B\rangle \in P$) if there exists a set $C \in P$ such that $A \subseteq C$ and $B \subseteq \overline{C}$.*

*(b) For any two pairs $\langle A, B\rangle$ and $\langle A', B'\rangle$, we say that $\langle A, B\rangle$ is G-*reducible *to $\langle A', B'\rangle$ if there is a polynomial-time computable function f such that $f(A) \subseteq A'$*

and $f(B) \subseteq B'$. Let $\mathcal{C}$ be a complexity class. We say that $\langle A, B\rangle$ is $\mathcal{C}$-hard if there exists a set C that is $\mathcal{C}$-hard and $\langle C, \overline{C}\rangle$ is G-reducible to $\langle A, B\rangle$.

It is clear that if $P \neq \mathcal{C}$ and that $\langle A, B\rangle$ is $\mathcal{C}$-hard, then $\langle A, B\rangle$ is not polynomial-time separable.

For any functions $s, l : \{0,1\} \to \mathbf{Q}^+$ such that $s(x) < l(x)$, we write $\langle f : l(x), s(x)\rangle$ to dentoe the pair of sets $\langle\{x \mid f(x) \geq l(x)\}, \{x \mid f(x) \leq s(x)\}\rangle$. The following proposition relates the hardness of approximating functions to that of pairs of decision problems.

Proposition 7 *Let $c : \mathbf{N} \to \mathbf{Q}^+$, $c(n) > 1$ for all $n \geq 0$, be polynomial-time computable. Let $s, l : \{0,1\}^* \to \mathbf{Q}^+$ be two polynomial-time computable functions satisfying $c(|x|)s(x) < l(x)/c(|x|)$. If $\langle f : l(x), s(x)\rangle$ is not polynomial-time separable, then the c-approximation problem of f is not computable in polynomial time.*

Proof. Assume that g is a function c-approximating f. Then, for any x, $|x| = n$, if $f(x) \geq l(x)$, then $g(x) > l(x)/c(n)$; if $f(x) \leq s(x)$, then $g(x) < c(n)s(x) < l(x)/c(n)$. Thus, from $g(x)$ and $l(x)/c(|x|)$, we can tell an instance in $\{x : f(x) \geq l(x)\}$ from an instance in $\{x : f(x) \leq s(x)\}$. □

Based on the above proposition, we define the notion of hardness of C-approximation problems as follows:

Definition 8 *Let $f : \{0,1\}^* \to \mathbf{Q}^+$ be a given function and $c : \mathbf{N} \to \mathbf{Q}^+$, $c(n) > 1$ be a polynomial-time computable function. We say that the c-approximation problem of f is $\mathcal{C}$-hard if there exist polynomial-time computable functions $s, l : \{0,1\}^* \to \mathbf{Q}^+$, $s(x) < l(x)$, such that*

1. *for all x of length n, $c(n)s(x) < l(x)/c(n)$; and*
2. *$\langle f : l(x), s(x)\rangle$ is $\mathcal{C}$-hard.*

We say the c-approximation problem of MINMAX-A is Π_2^P-complete if the decision version of MINMAX-A is in Π_2^P and the c-approximation problem of $f_{\text{MINMAX-A}}$ is Π_2^P-hard.

Remark. In practice, we often prove $\mathcal{C}$-hardness of $\langle f : l(x), s(x)\rangle$, where $l(x) = l \cdot \text{size}(x)$, $s(x) = s \cdot \text{size}(x)$ for some simple function size(x) and constants $0 < s < l \leq 1$. From Proposition 7, it follows that the $(l/s)^{1/2}$-approximation problem for f is $\mathcal{C}$-hard. The function size(x) is not necessarily the natural size of the instance x. Rather, it is a measure designed to prove the hardness of approximation. In the case of MINMAX-SAT, a 3-CNF boolean formula F has size$(F) = |F|$, the number of clauses in F.

The recent breakthrough of Arora et al [1] on the *NP*-hardness of many optimixation problems in the form of MAX-A is based on a characterization of *NP* in terms of the notion of *probabilistically checkable proofs* (*PCP*). Through a generalization of the notion of *PCP*, this characterization has been extended to the class Π_2^P (it was implicit in [2], and explicit in [6] and [5]). A consequence of this characterization is that the c-approximation problem of MINMAX-SAT is Π_2^P-complete for some constant $c > 0$. This will be our basis for proving other Π_2^P-complete c-approximation problems.

Proposition 9 *There exists a constant $0 < \epsilon < 1$ such that $\langle f_{\text{SAT}} : |F|, (1-\epsilon)|F| \rangle$ is Π_2^P-hard. Therefore, there is a constant $c > 1$ such that the c-approximation problem for f_{SAT} is Π_2^P-complete.*

5. Nonapproximability Results

Our proofs of the Π_2^P-completeness results for c-approximation problems MINMAX-A will be done by G-reductions from MINMAX-SAT to MINMAX-A. More precisely, we will construct G-reductions from $\langle f_{\text{SAT}} : |F|, (1-\epsilon)|F| \rangle$ to a pair $\langle f_{\text{MINMAX-A}} : (1-\epsilon_2)\text{size}(x), (1-\epsilon_1)\text{size}(x) \rangle$, where $\epsilon_1 > \epsilon_2 \geq 0$. For the proofs below, ϵ_2 are always 0. However, in [7], the G-reductions from MINMAX-SAT to MINMAX-SAT-B and LDC have $\epsilon_2 > 0$.

We first present the proof that the decision version of the problem MINMAX-CLIQUE is Π_2^P-complete. The result on the approximation version follows as a corollary.

Theorem 10 MINMAX-CLIQUE *is Π_2^P-complete.*

Proof. We reduce the problem SAT_2 to MINMAX-CLIQUE. Let F be a 3-CNF formula over variables $X = \{x_1, \ldots, x_r\}$ and $Y = \{y_1, \ldots, y_s\}$. Let $F = C_1 \wedge \ldots \wedge C_n$, where each C_i is the OR of three literals. We may assume that each clause C_i has at most one literal in $\{x_1, \ldots, x_r, \bar{x}_1, \ldots, \bar{x}_r\}$ (called *X-literals*). Otherwise, we can convert a clause C_i of two X-literals to two clauses each with one X-literal without changing the membership in SAT_2. For instance, if $C_i = x_1 \vee x_2 \vee y_1$, then we replace C_i with $C_{i,1} = x_1 \vee y_1 \vee z$ and $C_{i,2} = x_2 \vee y_1 \vee \bar{z}$, where z is a new variable not in $X \cup Y$, and it can be checked that the resulting formula $F'(X, Y \cup \{z\})$ is in SAT_2 if and only if $F(X, Y)$ is in SAT_2. (A clause C_i with 3 X-literals is trivially false for some τ_1.) We now describe the construction of the graph G.

The vertex set V of G is partitioned into $2n$ subsets $V_{i,j}$, $1 \leq i \leq n$, $j = 0, 1$ (i.e., $I = n, J = 2$). For each $1 \leq i \leq n$ and $0 \leq j \leq 1$, $V_{i,j}$ has 3 vertices $a_{i,j}[k]$, $k = 1, 2, 3$, corresponding to the 3 literals of C_i, together with $n' = \lceil n/2 \rceil$ other vertices $b_{i,j}[k]$, $k = 1, \ldots, n'$. Let $B_{i,j} = \{b_{i,j}[k] : k = 1, \ldots, n'\}$. Within $B_{i,j}$, all vertices are connected (and so $B_{i,j}$ is a clique of size n'). There is no other edge within $V_{i,j}$, and there is no edge between $V_{i,0}$ and $V_{i,1}$.

To define the edges between the vertices in $V_{i,j}$ and vertices in $V_{i',j'}$ with $i \neq i'$, we associate an X-literal $xl(V_{i,j})$ to each $V_{i,j}$. Each C_i has at most one X-literal. Suppose C_i has an X-literal; then we let $xl(V_{i,1})$ be the literal in C_i and $xl(V_{i,0})$ be its complement. Suppose C_i has no X-literals; then we add a dummy variable x_{r+1} and let $xl(V_{i,0}) = xl(V_{i,1}) = x_{r+1}$. In addition, we define the following terms on vertices $a_{i,j}[k]$: A vertex $a_{i,j}[k]$ is *negative* if it corresponds to a X-literal in C_i and $j = 0$. Two vertices $a_{i,j}[k]$ and $a_{i',j'}[k']$ are *complementary* if they correspond to two complementary literals, i.e., one is x_k (or, y_k) and the other is $\bar{x}_k$ (or, respectively, $\bar{y}_k$) for some k.

Now, we define edges between $V_{i,j}$ and $V_{i',j'}$ with $i \neq i'$ as follows:

(1) If $xl(V_{i,j})$ and $xl(V_{i',j'})$ are complementary, then we connect all vertices between $B_{i,j}$ and $B_{i',j'}$. There is no other edge between $V_{i,j}$ and $V_{i',j'}$.

(2) If not (1), then there is no edge between $B_{i,j}$ and $V_{i',j'}$ and no edge between $B_{i',j'}$ and $V_{i,j}$. For any two vertices $a_{i,j}[k]$ and $a_{i',j'}[k']$, they are connected by an edge if and only if they are nonnegative and noncomplementary.

The above completes the graph G. Now we prove that this construction is correct with respect to the bound $K = n$. It suffices to prove the following stronger statement:

Claim. If $f_{\mathrm{SAT}}(F) > \lceil |F|/2 \rceil$, then $f_{\mathrm{CLIQUE}}(G) = f_{\mathrm{SAT}}(F)$.

Proof of Claim. First assume that for each truth assignment τ_1 on X, there is a truth assignment τ_2 on Y that satisfies k^* clauses of F (i.e., $f_{SAT}(F) = k^*$). Let t be any mapping from $\{1, \ldots, n\}$ to $\{0, 1\}$. First, if for some $i \neq i'$, $xl(V_{i,t(i)})$ and $xl(V_{i',t(i')})$ are complementary, then we get a clique $B_{i,t(i)} \cup B_{i',t(i')}$ that is of size $\geq n \geq k^*$. Second, if for all $i \neq i'$, $xl(V_{i,t(i)})$ and $xl(V_{i',t(i')})$ are noncomplementary, then there is a unique truth assignment τ_1 on X such that $\tau_1(xl(V_{i,t(i)})) = 1$ for all i, $1 \leq i \leq n$. (We always let τ_1 on the dummy variable x_{r+1} be 1.) For this assignment τ_1, there is a truth assignment τ_2 on Y that satisfies k^* clauses. Let $I_T = \{i : C_i(\tau_1, \tau_2) = 1\}$. For each $i \in I_T$, pick a vertex $a_{i,t(i)}[k]$ that corresponds to a *true* literal in C_i under τ_1 and τ_2. Note that if we picked a vertex $a_{i,t(i)}[k]$ that corresponds to the X-literal in C_i, then $t(i)$ must be equal to 1. Thus, it is easy to check that these vertices $a_{i,t(i)}[k]$ form a clique of size k^*: (i) if $a_{i,t(i)}[k]$ corresponds to an X-literal, then as observed above $t(i) = 1$ and it is nonnegative; (ii) no two selected vertices are complementary, since the corresponding literals must be noncomplementary to be satisfied by τ_1 and τ_2.

Conversely, assume that the maximum clique size of all G_t for all $t : \{1, \ldots, n\} \to \{0, 1\}$ is at least k^*. Let τ_1 be any truth assignment on X. We need to show that there is a truth assignment τ_2 on Y that satisfies k^* clauses. Define a mapping $t : \{1, \ldots, n\} \to \{0, 1\}$ by $t(i) = 1$ if and only if $\tau_1(xl(V_{i,1})) = 1$, i.e., $\tau_1(xl(V_{i,t(i)})) = 1$ for all $i \leq n$ (we assume that $\tau_1(x_{r+1}) = 1$). Thus, no two X-literals $xl(V_{i,t(i)})$ and $xl(V_{i',t(i')})$ are complementary, and so there is no edge between $B_{i,t(i)}$ and $V_{i',t(i')}$ if $i \neq i'$. It follows that the maximum clique Q of G_t must consist of a single vertex $a_{i,t(i)}[k]$ in $V_{i,t(i)}$, for k^* indices i. Let $I_Q = \{i : Q \cap V_{i,t(i)} \neq \emptyset\}$. Now, we define a truth assignment τ_2 on Y as follows: if y_ℓ ever occurs as a literal corresponding to some vertex in the clique Q, then assign $\tau_2(y_\ell) = 1$; otherwise, assign $\tau_2(y_\ell) = 0$. We check that τ_1 and τ_2 satisfy C_i for all $i \in I_Q$. In particular, for each $i \in I_Q$, the literal corresponding to the vertex $a_{i,t(i)}[k]$ in $V_{i,t(i)} \cap Q$ must be true to τ_1 and τ_2:

(1) If $a_{i,t(i)}[k]$ corresponds to an X-literal and it belongs to the clique Q, then it must be nonnegative and so $t(i) = 1$. That means the corresponding X-literal is the same as $xl(V_{i,1})$ and has the value 1 under τ_1.

(2) If $a_{i,t(i)}[k]$ corresponds to a Y-literal y_ℓ, then since y_ℓ occurs in Q, $\tau_2(y_\ell) = 1$.

(3) If $a_{i,t(i)}[k]$ corresponds to a Y-literal $\bar{y}_\ell$, then y_ℓ does not occur in the clique Q, because y_ℓ and $\bar{y}_\ell$ are complementary and so they cannot be connected. This implies that $\tau_2(\bar{y}_\ell) = 1$.

The above completes the proof of the claim and hence the correctness of the reduction. □

Corollary 11 *There exists a constant $c > 1$ such that the c-approximation problem of* MINMAX-CLIQUE *is Π_2^P-complete.*

Proof. Let g be the reduction of the above theorem. In the above proof, we showed that for any 3-CNF formula F, $f_{\text{SAT}}(F) = f_{\text{CLIQUE}}(g(F))$ as long as $f_{\text{SAT}}(F) > \lceil |F|/2 \rceil$. For any graph G whose vertex set V is partitioned into $V_{i,j}$, $1 \le i \le I$, $1 \le j \le J$. We let $\text{size}(G) = I$. Then the above observation implies that g is a G-reduction from $\langle f_{\text{SAT}} : |F|, (1-\epsilon)|F| \rangle$ to $\langle f_{\text{CLIQUE}} : \text{size}(G), (1-\epsilon)\text{size}(G) \rangle$. □

We note that the Π_2^P-completeness of MAXMIN-VC follows from that of MINMAX-CLIQUE since they are the dual problems to each other. However, the c-approximation results do not carry over.

Corollary 12 MAXMIN-VC *is* Π_2^P*-complete.*

Next, we prove that MINMAX-3DM and its c-approximation version are Π_2^P-complete. In order to do this, we need the Π_2^P-completeness result on a stronger version of the problem MINMAX-SAT.

MINMAX-SAT-YB: given $F(X, Y)$, with the number of occurrences of each $y \in Y$ bounded by a constant b, find $f_{\text{SAT-YB}}(F) = f_{\text{SAT}}(F)$.

The more general case of MINMAX-SAT-B, in which the number of occurrences of all variables in X or Y are bounded, is also Π_2^P-complete. Its proof is more involved and is given in a separate paper [7]. Here we give a sketch for the Π_2^P-completeness of this simpler case MINMAX-SAT-YB.

Theorem 13 MINMAX-SAT-YB *is* Π_2^P*-complete.*

Proof. (Sketch) The proof is a simple modification of the reduction from the maximum satisfiability problem to the bounded-occurrence maximum satisfiability problem in [10]. It was shown in [10] that there exist a polynomial-time computable function f and two integers $\alpha, b > 0$ such that

(i) for each 3-CNF formula $F(X)$ with m clauses, $f(F(X)) = F'(X')$ is a 3-CNF boolean formula with $(\alpha+1)m$ clauses in which each variable occurs at most b times, and

(ii) $\max_{\tau' : X' \to \{0,1\}} tc(F'(\tau')) = \alpha m + \max_{\tau : X \to \{0,1\}} tc(F(\tau))$,

Now, for each $F(X, Y)$, we treat all $x \in X$ as constants and compute $f(F(X, Y)) = F'(X, Y')$. Then, we have

$$\begin{aligned} \min_{\tau_1 : X \to \{0,1\}} \max_{\tau_2' : Y' \to \{0,1\}} tc(F'(\tau_1, \tau_2')) &= \min_{\tau_1 : X \to \{0,1\}} \left[\alpha m + \max_{\tau_2 : Y \to \{0,1\}} tc(F(\tau_1, \tau_2)) \right] \\ &= \alpha m + \min_{\tau_1 : X \to \{0,1\}} \max_{\tau_2 : Y \to \{0,1\}} tc(F(\tau_1, \tau_2)). \end{aligned}$$

Thus, if $\langle f_{\text{SAT}} : |F|, (1-\epsilon)|F| \rangle$ is Π_2^P-hard, then $\langle f_{\text{SAT-YB}} : |F|, (1-\epsilon/(\alpha+1))|F| \rangle$ is also Π_2^P-hard. □

Theorem 14 MINMAX-3DM *is* Π_2^P*-complete.*

Proof. We will construct a G-reduction from MINMAX-SAT-YB to MINMAX-3DM. It is a modification of the L-reduction from the maximum satisfiability problem with

bounded occurrences of variables to the maximum 3DM problem [4]. We first give a brief review of that proof.

Let $F(X) = C_1 \wedge \ldots \wedge C_n$ be a 3-CNF boolean formula over variables $Y = \{y_1, \ldots, y_s\}$. Let d_i be the number of occurrences of y_i or $\bar{y}_i$ in F. (Assuming that each clause $C_\ell$has exactly 3 literals, we have $\sum_{i=1}^{s} d_i = 3n$.) We assume that $d_i \leq b$ for all $i = 1, \ldots, s$. Let M be the minimum number greater than $3b/2 + 1$ such that M is a power of 2. We describe below a collection S of 3-element subsets of a set W (called *triples*), without explicitly writing down all the names of elements in W.

(1) For each variable y_i, define M identical sets of *ring triples*. For each y_i and each k, $1 \leq k \leq M$, the ring $R_{i,k}$ contains two sets of triples:

$$\begin{aligned} R^1_{i,k} &= \{\{\bar{y}_i[j,k], a_i[j,k], b_i[j,k]\} : 1 \leq j \leq d_i\}, \\ R^0_{i,k} &= \{\{y_i[j,k], b_i[j,k], a_i[j+1,k]\} : 1 \leq j \leq d_i\}, \end{aligned}$$

where $j+1$ in $a_i[j+1,k]$ is the addition modulo d_i. This ring is the basic component of the reduction from SAT to 3DM in, e.g., [3]. We show it in Figure 6(a). Here $a_i[j,k]$ and $b_i[j,k]$ are local elements and appear in no other triples. Thus, any complete matching must take all triples in $R^1_{i,k}$ or all triples in $R^0_{i,k}$, corresponding to setting all occurrences of y_i true or all false.

(2) For each variable y_i and each j, $1 \leq j \leq d_i$, construct two sets of *tree triples*: $T^0_{i,j}$ and $T^1_{i,j}$. Set $T^1_{i,j}$ forms a tree of size $2M-1$ with leaves $y_i[j,k]$, $1 \leq k \leq M$, and set $T^0_{i,j}$ with leaves $\bar{y}_i[j,k]$, $1 \leq k \leq M$. We show a tree $T^1_{i,j}$ in Figure 6(b). All the internal nodes of the trees, except the roots, are the local elements. The root of $T^1_{i,j}$ is called $u_i[j]$ and the root of $T^0_{i,j}$ is called $\bar{u}_i[j]$. It is shown in [4] that, a maximum matching must take, for any i and j, all $y_i[j,k]$'s by ring triples or all by tree triples. For instance, in Figure 6, with $d_i = 4$ and $M = 8$, a maximum matching must take all circled triples or all noncircled triples. In other words, for each i, $1 \leq i \leq s$, the maximum matching will match all ring triples and tree triples so that only all $u_i[j]$'s are left free or only all $\bar{u}_i[j]$'s are left free. If the matching leaves $u_i[j]$'s free, then we say it corresponds to the truth assignment that sets y_i true. For instance, the matching taking all circled triples in Figure 6 corresponds to assigning y_i false.

(3) Identify $u_i[j]$ with the jth occurrence of y_i, and identify $\bar{u}_i[j]$ with the jth occurrence of $\bar{y}_i$. For each clause C_ℓ, define 3 *clause triples*: $\{s_1[\ell], s_2[\ell], w\}$, where w ranges over the three roots of the trees $T^0_{i,j}$ and $T^1_{i,j}$ that are identified as above to the three literals in C_ℓ.

(4) Define *garbage triples* $\{g_{2q-1}, g_{2q}, u_i[j]\}$ and $\{g_{2q-1}, g_{2q}, \bar{u}_i[j]\}$ for all $q = 1, \ldots, 2n$ and all $i = 1, \ldots, s$ and all $j = 1, \ldots, d_i$.

The above are all triples. Some simple calculation shows that there are totally $18nM$ elements in W and the number of matching is at most $6nM$ that covers all elements. We let $K = 6nM$.

We now show that the reduction is correct. From the remarks in (1) and (2) above, we see that the maximum matchings on ring triples and tree triples correspond one-to-one with the truth assignments on Y. So, if F is satisfied by a truth assignment τ on Y, then we select disjoint triples from ring triples and tree triples so that for each y_i with $\tau(y_i) = 1$, only the nodes $u_i[j]$'s are left free, and for each y_i with $\tau(y_i) = 0$, only the nodes $\bar{u}_i[j]$'s are left free. It can be checked that there are $(6M-3)n$ such triples. For each clause C_ℓ, we select the clause triple that covers

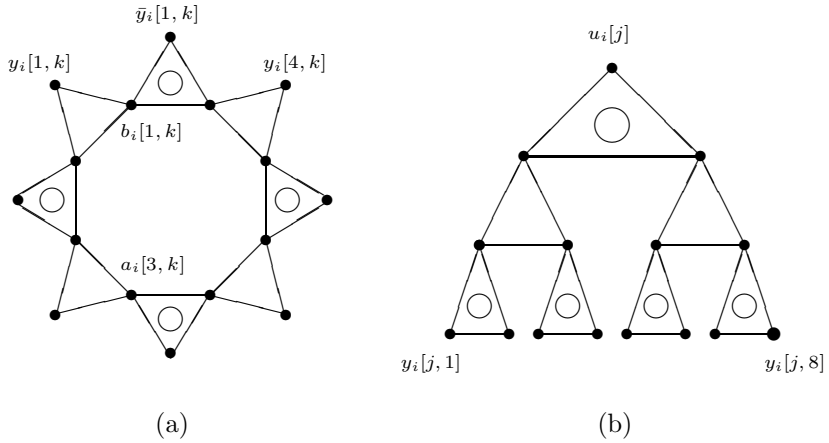

Fig. 6. (a) Ring triples with $d_i = 4$. (b) Tree triples with $M = 8$. Each triangle denotes a triple.

the root that corresponds to a true literal in C_ℓ. Finally, we cover all other roots by garbage clauses. This is a complete matching of size $6nM$.

Conversely, if there is a matching that covers every element, then it must contain for each clause C_ℓ a clause triple $\{s_1[\ell], s_2[\ell], w\}$, where w is a free root note and also corresponds to a literal in C_ℓ. By the property of the maximum matchings discussed in (2) above, we can define truth assignment τ on Y to make all such literals true and so to satisfy F.

Now we describe our modification for MINMAX-3DM. First, we divide W into n groups $W_1, \ldots, W_n$, with each W_ℓ containing all elements of W that occur in the ring triples and tree triples related to clause C_ℓ. More specifically, suppose the jth occurrence of y_i or $\bar{y}_i$ is in C_ℓ, then W_ℓ contains all elements in the trees $T^0_{i,j}$ and $T^1_{i,j}$, plus the internal elements $a_i[j,k]$ and $b_i[j,k]$ for all $k \le K$. In addition, W_ℓ contains $s_1[\ell], s_2[\ell]$ and $g_{4\ell-p}$, $p = 0, 1, 2, 3$. For each ℓ, $1 \le \ell \le n$, we have some local triples that contain only elements in W_ℓ and inter-group triples that contain some elements in W_ℓ and some not in W_ℓ. For instance, all tree triples and clause triples are local, some ring triples are local and some ring triples and garbage triples are inter-group.

Now, suppose $F(X,Y) = C_1 \wedge \ldots \wedge C_n$ is a 3-CNF formula over two variable sets $X = \{x_1, \ldots, x_r\}$ and $Y = \{y_1, \ldots, y_s\}$, with each variable y_i of Y occurring in F at most b times. As explained in the proof of Theorem 10, we may assume that each clause C_ℓ contains at most one X-literal. We treat the variables in X as constants, and define the triples as above from F, and divide them into groups W_ℓ, $\ell = 1, \ldots, n$. (Note that for each clause C_ℓ with an X-literal, it has only 2 clause triples of the form $\{s_1[\ell], s_2[\ell], w\}$.) Next, for each $1 \le \ell \le n$ and each $m = 0, 1$, we define $W_{\ell,m}$ to be a copy of W_ℓ ; i.e., for each element in W_ℓ, attach an additional index m to it (so, e.g., $s_1[\ell]$ becomes $s_1[\ell, 0]$ in $W_{\ell,0}$). Then, for each group $W_{\ell,m}$, we add elements $\alpha_{\ell,m}[k], \beta_{\ell,m}[k], \gamma_{\ell,m}[k]$, for $k = 1, \ldots, n$. If C_ℓ has an X-literal which is positive, we add one more element σ_ℓ to $W_{\ell,1}$, else we add it to $W_{\ell,0}$. We define the set S' as follows:

(1) For each local triple in W_ℓ, we include its *copies* in both $W_{\ell,0}$ and $W_{\ell,1}$ in S'.

(2) For each inter-group triple between W_ℓ and $W_{\ell'}$, we include all its *copies* between $W_{\ell,m}$ and $W_{\ell',m'}$, for all $m, m' = 0, 1$, in S'.

(3) For each C_ℓ, if x_i is a literal of C_ℓ, then add a triple $\{s_1[\ell, 1], s_2[\ell, 1], \sigma_\ell\}$ to S'; if $\bar{x}_i$ is a literal of C_ℓ, then add a triple $\{s_1[\ell, 0], s_2[\ell, 0], \sigma_\ell\}$ to S';

(4) We say two pairs (ℓ, m) and (ℓ', m') are *inconsistent* if both C_ℓ and $C_{\ell'}$ have the same X-literal but $m \neq m'$ or if C_ℓ and $C_{\ell'}$ have the complementary X-literal but $m = m'$. If (ℓ, m) and (ℓ', m') are inconsistent, then we add the triples $\{\alpha_{\ell,m}[k], \beta_{\ell,m}[k], \gamma_{\ell',m'}[k]\}$ to S' for all $k = 1, \ldots, n$.

Finally, we let $K = 6nM$, and claim that the reduction is correct.

First, assume that $F(X, Y) \in \text{SAT}_2$, and let t be a function from $\{1, \ldots, n\}$ to $\{0, 1\}$. We check that there are at least $6nM$ matchings in $W_t = \bigcup_{\ell=1}^{n} W_{\ell,t(\ell)}$. First, as in the original reduction (from SAT to 3DM), we can select $(6M - 1)n$ disjoint triples from ring triples, tree triples and garbage triples. Suppose for some ℓ, ℓ', $(\ell, t(\ell))$ and $(\ell', t(\ell'))$ are inconsistent. Then, we can get from (4) above at least n disjoint triples to make a matching of at least $6nM$ triples. Suppose t is consistent. Then, it defines a truth assignment τ_1 on X and for this τ_1 there is a truth assignment τ_2 on Y satisfying F. It follows from the analysis of the original reduction that there is a matching of $6nM$ triples. Note that for each clause C_ℓ if τ_1 satisfies C_ℓ, then the corresponding $W_{\ell,t(\ell)}$ must contain σ_ℓ and $\{s_1[\ell, t(\ell)], s_2[\ell, t(\ell)], \sigma_\ell\}$ must be in S'.

Conversely, if $F(X, Y) \notin \text{SAT}_2$ then there exists a truth assignment τ_1 on X such that $F(\tau_1, Y)$ is not satisfiable. Choose the corresponding t, i.e., $t(\ell) = 1$ if and only if τ sets the X-literal in C_ℓ true. This function t must be consistent and so there is no triple from the extra elements such as $\alpha_{\ell,m}[k]$. The only triples are the *copies* of those in the original reduction, and there are less than $6nM$ disjoint triples. □

Corollary 15 *There exists a constant $c > 1$ such that the c-approximation problem for* MINMAX-3DM *is Π_2^P-complete.*

Proof. We observe that the original reduction (from SAT to 3DM), preserves the optimum solution in the following sense: if the maximum number of satisfiable clauses is β, then the maximum matching has $(6M - 1)n + \beta n$ triples [4]. The main idea was that the design of the tree triples forces the maximum matching to make consistent truth assignments to the different occurrences of y_i. In the new reduction, this property is preserved if the function t is consistent. (If t is not consistent, then there are always at least $6nM$ disjoint triples.)

For each instance (W, S) of MINMAX-3DM with W partitioned into subsets $W_{\ell,m}$, with $1 \leq \ell \leq I$, $1 \leq m \leq J$, let $\text{size}(W, S) = 6MI$. Then, the above observation shows that the new reduction is a G-reduction from $\langle f_{\text{SAT}-\text{YB}} : |F|, (1 - \epsilon)|F| \rangle$ to $\langle f_{\text{3DM}} : \text{size}(W, S), (1 - \epsilon/6M)\text{size}(W, S) \rangle$. □

6. Conclusion and Open Questions

We have demonstrated a number of min-max optimization problems to be Π_2^P-complete. Using the idea of parameterized inputs, there are apparently many more

similar results on the generalization of *NP*-complete problems. For instance, the Π_2^P-completeness results also hold for the generalized knapsack problem and the generalized maximum set covering problem. It is hoped that these new Π_2^P-completeness results are useful for proving other natural problems such as GRN to be complete for Π_2^P or Σ_2^P.

Although the Π_2^P-completeness results for the min-max optimization problems appear easy to prove, the corresponding Π_2^P-completeness results for the c-approximation problems are harder to get. We were successful only for a few such problems. It would be interesting to develop techniques for classifying the complexity of the c-approximation problem of the min-max problems (like the class MAX SNP for problems of the form MAX-A). In particular, it would be interesting to know whether the c-approximation problems of f_{CIRCUIT} and f_{VC} are Π_2^P-complete.

References

1. A. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy, Proof verification and hardness of approximation problems, *Proceedings, 33rd IEEE Symposium on Foundations of Computer Science* (1992), 14-23.
2. A. Condon, J. Feigenbaum, C. Lund and P. Shor, Probabilistically checkable debate systems and approximation algorithms for PSPACE-hard functions, *Proceedings, 25th ACM Symposium on Theory of Computing* (1993), 305-314.
3. M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.
4. V. Kann, Maximum bounded 3-dimensional matching is MAX SNP-complete, *Inform. Proc. Lett.* 37 (1991), 27-35.
5. M. Kiwi, C. Lund, A. Russell, D. Spielman and R. Sundaram, Alteration in Interaction, *Proceedings, 9th Structure in Complexity Theory Conference*, IEEE (1994), 294-303.
6. K. Ko and C.-L. Lin, Non-approximability in the polynomial-time hierarchy, *Tech. Report TR-94-2*, Department of Computer Science, State University of New York at Stony Brook, Stony Brook, 1994.
7. K. Ko and C.-L. Lin, On the longest circuit in an alterable digraph, preprint, 1994.
8. K. Ko and W.-G. Tzeng, Three Σ_2^P-complete problems in computational learning theory, *comput. complex.* 1 (1991), 269-301.
9. C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, New Jersey, 1982.
10. C. H. Papadimitriou and M. Yannakakis, Optimization, approximation, and complexity classes, *J. Comput. System Sci.* 43 (1991), 425-440.
11. C. H. Papadimitriou and M. Yannakakis, The traveling salesman problem with distances one and two, *Math. Oper. Res.* 18 (1993), 1-11.
12. Y. Sagiv and M. Yannakakis, Equivalences among relational expressions with the union and difference operations, *J. Assoc. Comput. Mach.* 27 (1980), 633-655.
13. L. J. Stockmeyer, The polynomial-time hierarchy, *Theoret. Comput. Sci.* 3 (1977), 1-22.
14. K. W. Wagner, The complexity of combinatorial problems with succinct input representation, *Acta Inform.* 23 (1986), 325-356.