The uniform minimum-ones 2SAT problem and its application to haplotype classification

The Uniform Minimum-Ones 2SAT Problem and its Application to Haplotype Classification*

Hans-Joachim Böckenhauer[1], Ján Oravec[1],
Björn Steffen[1], Kathleen Steinhöfel[2], and Monika Steinová[1]

[1] Department of Computer Science, ETH Zurich, Switzerland,
{hjb, oravecj, bjoern.steffen, monika.steinova}@inf.ethz.ch
[2] Department of Computer Science, King's College London, UK
kathleen.steinhofel@kcl.ac.uk

Abstract. Analyzing genomic data for finding those gene variations which are responsible for hereditary diseases is one of the great challenges in modern bioinformatics. In many living beings (including the human), every gene is present in two copies, inherited from the two parents, the so-called *haplotypes*. In this paper, we propose a simple combinatorial model for classifying the set of haplotypes in a population according to their responsibility for a certain genetic disease. This model is based on the minimum-ones 2SAT problem with uniform clauses.
The minimum-ones 2SAT problem asks for a satisfying assignment to a satisfiable formula in 2CNF which sets a minimum number of variables to true. This problem is well-known to be $\mathcal{NP}$-hard, even in the case where all clauses are uniform, i. e., do not contain a positive and a negative literal. We analyze the approximability and present the first non-trivial exact algorithm for the uniform minimum-ones 2SAT problem with a running time of $\mathcal{O}(1.25993^n)$ on a 2SAT formula with n variables.

1 Introduction

With the enhancement of sequencing methods, more and more genetic data becomes available for research. One of the grand challenges in molecular biology is to interpret this genomic data and to make use of it, for example for the detection of genetic diseases. In this paper, we investigate a challenging combinatorial problem which arises in this context.

Humans and many other living organisms (including all vertebrates) have a diploid genome, i. e., they have two copies for each chromosome. For an individual organism, these two copies usually differ from each other as a result of their inheritance from two separate parent entities. The most common differences are the so-called single nucleotide polymorphisms (SNPs), where a single nucleotide is replaced by another one. For a given chromosome, the complete sequence information of one copy is called a *haplotype*.

For the interpretation of such haplotype data, we consider the following classification problem: Assume that the haplotype data of a population is given, together

* This work was partially supported by SNF grant 200021-109252/1.

with some phenotypical data describing whether the individuals have symptoms of a specific genetically influenced disease or not. Our goal is to find a plausible subset of bad haplotypes occurring in the population which are responsible for this disease. We employ a very simple model by assuming that every ill individual carries at least one bad haplotype and that every healthy individual carries at least one good haplotype.

This problem can be formally described as the following variant of a satisfiability problem: We are given a satisfiable formula in 2CNF and the goal is to compute a satisfying assignment which minimizes the number of variables set to true. This problem is known as the *minimum-ones 2SAT problem*, it was first considered by Gusfield and Pitt [5] who presented a 2-approximation algorithm for it. Although it is possible to decide the satisfiability of a 2CNF formula in linear time [1], the minimum-ones 2SAT problem is $\mathcal{NP}$-hard and even $\mathcal{APX}$-hard since it is a generalization of the vertex cover problem.

In this paper, we concentrate on the *uniform* variant of the minimum-ones 2SAT problem, where the input formula does not contain any clause consisting of a negated and an unnegated variable. This problem variant exactly models our haplotype classification problem. This variant is hard as well, so we analyze its approximability and design an exact algorithm with a moderately exponential running time of $\mathcal{O}(1.25993^n)$ for it.

The paper is organized as follows. In Section 2, we fix our notation and give a formal definition of the minimum-ones 2SAT problem. In Section 3, we describe how to apply our result to the haplotype classification problem. Section 4 is devoted to the approximability of the minimum-ones 2SAT problem. In Section 5, we present our main result, namely, a fast exact algorithm for the uniform case of the minimum-ones 2SAT problem. We conclude the paper in Section 6. Due to space limitations, some proofs are moved to the appendix.

2 Preliminaries

In this section, we define the basic notions we will be using throughout the paper. We consider Boolean formulas over a set of variables $X = \{x_1, \ldots, x_n\}$. A *literal* is either some variable $x \in X$ or its negation $\overline{x}$. A *clause* is a disjunction of literals, i. e., a formula of the form $(y_1 \vee y_2 \vee \cdots \vee y_k)$ where the y_i are literals over X. A Boolean formula is said to be in *conjunctive normal form (CNF)* if it is a conjunction of disjunctions of literals, i. e., a formula $C_1 \wedge \ldots \wedge C_m$ for some clauses $C_1, \ldots, C_m$. We say that a CNF formula is in *2CNF* if every clause contains exactly 2 literals. A 2CNF clause is called *positive*, if both literals in it are unnegated variables, it is called *negative*, if both literals are negated variables, and *mixed*, if it contains one negated and one unnegated variable.

An *assignment* for a formula φ over X is a function $\beta\colon X \to \{0, 1\}$ assigning a truth value to every variable. An assignment β *satisfies* a formula φ, if the standard Boolean evaluation of φ using the assignment β yields the value 1. A formula is *satisfiable*, if there exists a satisfying assignment for it.

Definition 1. Min-Ones-2SAT *is the following optimization problem:*

Input: *A satisfiable 2CNF formula* $\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ *over the set of variables* $X = \{x_1, \ldots, x_n\}$.
Feasible solutions: *All satisfying assignments for* φ.
Costs: *The cost of a satisfying assignment* β *is the number of variables set to* 1 *by* β.
Goal: *Minimization.*

Uniform Min-Ones-2SAT *is the restriction of* Min-Ones-2SAT *to input instances without mixed clauses, and* Positive Min-Ones-2SAT *is the restriction of* Uniform Min-Ones-2SAT *to inputs containing only positive clauses.*

For our algorithms, we are going to use a graph representation for the Uniform Min-Ones-2SAT where the vertices of the graph correspond to the variables and the edges correspond to the clauses.

Definition 2. *For a formula* φ *in 2CNF without mixed clauses, we define the* satisfiability graph $G_\varphi = (X, E_+ \cup E_-)$*, where* $E_+ = \{\{x, y\} \mid (x \vee y)$ *is a positive clause in* $\varphi\}$ *and* $E_- = \{\{x, y\} \mid (\overline{x} \vee \overline{y})$ *is a negative clause in* $\varphi\}$*. We call the edges from* E_+ positive edges *and those from* E_- negative edges*. By* G_φ^+ *and* G_φ^-*, we denote the graph induced by the edges* E_+ *and* E_-*, respectively.*

Note that the satisfiability graph is actually a multigraph, since two vertices may be connected by both a positive and a negative edge. Recall that a *vertex cover* of a graph $G = (V, E)$ is a set $X \subseteq V$ of vertices such that every edge is incident to at least one vertex from X. An *independent set* of a graph $G = (V, E)$ is a set $X \subseteq V$ of vertices that are pairwise non-adjacent. The *minimum vertex cover problem* (Min-VC) is the problem of finding a vertex cover of minimum cardinality in a given graph, whereas the *maximum independent set problem* is the problem of finding an independent set of maximum cardinality in a given graph.

Observation 1 Positive Min-Ones-2SAT *is equivalent to* Min-VC.

Observation 2 Uniform Min-Ones-2SAT *is equivalent to the problem of finding a minimum vertex cover in* G_φ^+ *that is an independent set in* G_φ^-.

3 Application to a Haplotype Classification Problem

In this chapter, we will present the application of Uniform Min-Ones-2SAT for classifying haplotypes in more detail. The goal is to find genes which are responsible for the predisposition for some diseases.

Recall that the genome of every human is present in two copies in every cell. These two copies, called *haplotypes*, slightly differ from each other; one is inherited from the mother, the other from the father. In the following, we will not look at the whole genome of the individuals, but on smaller units, for example at the haplotypes of a single gene. This means, that, in a population of related individuals, some of the haplotypes will occur frequently, e. g., because they have been conserved over some generations.

Table 1. Types of individuals according to haplotype and phenotype data

type	phenotype	haplotypes
1	healthy	both normal
2	healthy	1 normal, 1 defective
3	symptoms of disease	1 normal, 1 defective
4	symptoms of disease	both defective
5	healthy	both defective
6	symptoms of disease	both normal

Some of these haplotypes might show genetic variations that are responsible for a predisposition for some kind of disease. It is of great importance to find these variations for a better understanding of such diseases and for the development of more successful treatments. Roughly speaking, our goal is to find these defective haplotypes in our data. The first problem is that the genomic data generated by many sequencing experiments does not produce the haplotypes of an individual, but rather a mixture of both haplotypes, the so-called *genotype*. It is a challenging problem by itself to compute the haplotypes from the genotype, but some successful algorithms have been presented in the literature, for an overview see, for example, [2, 11, 3].

We do not want to consider this problem here, rather we assume that we have access to some reliable haplotype data. Together with this haplotype data for some population of individuals, we are given some phenotypical data describing for each individual whether it shows some symptoms of the disease or not.

It is assumed that the predisposition for the disease is connected to some genetic defect on one or both haplotypes of the individual. The goal is to learn how to distinguish haplotypes with this defect from normal ones.

We now want to classify the individuals from the population into six different types according to the relationship between their haplotypes and the observed symptoms of the disease as shown in Table 1. Note that the information from the second column of the table is known to us whereas the information from the third column is what we would like to deduce.

For a first attempt of classifying the haplotypes and finding the defective ones, we want to simplify the model by restricting ourselves to the first four rows of Table 1. This means, we assume that the population does neither include individuals which show symptoms of the disease but have two normal haplotypes nor healthy individuals with two defective haplotypes. Intuitively, this means, we assume that an individual with two defective haplotypes will almost surely show symptoms of the disease, whereas it is rare for an individual to show symptoms of the disease without genetic predisposition.

This problem can now be modelled by Uniform Min-Ones-2SAT: Consider the haplotypes as the variables of the formula, where an assignment of 1 to a variable means to classify the respective haplotype as defective. Then every individual of the population describes one clause of the formula. For a healthy individual,

at least one of its two haplotypes has to be normal, this corresponds to a negative clause. On the other hand, if an individual shows symptoms of the disease, this means that at least one of its haplotypes has to be defective, corresponding to a positive clause. Therefore, finding an assignment with as few as possible ones, represents an explanation for the disease with the smallest possible number of defective haplotypes. This mirrors our expectation that the population should contain more normal haplotypes than defective ones.

4 Approximability of Uniform Min-Ones-2SAT

In this section, we present some results on the approximability of Uniform Min-Ones-2SAT. We start with an upper bound.

Observation 3 *There exists a polynomial-time approximation algorithm for* Uniform Min-Ones-2SAT *with ratio 2.*

Proof. Gusfield and Pitt have shown that Min-Ones-2SAT is 2-approximable [5]. Since Uniform Min-Ones-2SAT is a special case of Min-Ones-2SAT, the claim immediately follows. □

Since Uniform Min-Ones-2SAT is a generalization of Min-VC according to Observation 2, any approximation ratio better than $2 - \frac{1}{\Delta_G}$ (where Δ_G denotes the maximum degree of the satisfiability graph) or better than $2 - \frac{1}{\Theta(\sqrt{\log n})}$ would improve on the best known Min-VC approximation [9, 7].

Regarding the lower bounds on the approximation ratio, Uniform Min-Ones-2SAT inherits the results from Min-VC. This leads to the following observation.

Observation 4 Uniform Min-Ones-2SAT *is $\mathcal{APX}$-hard and not approximable within a ratio of* 1.3606*, unless $\mathcal{P} = \mathcal{NP}$ and not approximable within a factor better than 2 unless the unique games conjecture holds.*

Proof. This follows immediately from the respective result for Min-VC [4, 8]. □

We now prove that Uniform Min-Ones-2SAT is as hard to approximate as Min-Ones-2SAT.

Theorem 1. *For every polynomial-time α-approximation algorithm A_U for* Uniform Min-Ones-2SAT*, there exists a polynomial-time algorithm A_N for* Min-Ones-2SAT *with asymptotic approximation ratio α.*

Proof. Let A_U be a polynomial-time α-approximation algorithm for Uniform Min-Ones-2SAT. We show in the following, how A_U can be used to find an asymptotic α-approximate solution to any Min-Ones-2SAT instance.

Consider an input instance φ for Min-Ones-2SAT over the set of variables $X = \{x_1, \ldots, x_n\}$. Let m denote the number of clauses of φ and let m_N denote the

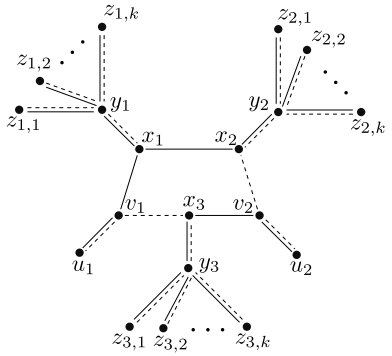

Fig. 1. The satisfiability graph of the uniform 2CNF formula ψ as constructed in the proof of Theorem 1 for the 2CNF formula $\varphi = (x_1 \vee \overline{x_3}) \wedge (\overline{x_2} \vee x_3) \wedge (x_1 \vee x_2)$. Solid lines denote positive edges, dashed lines denote negative edges.

number of non-uniform (i. e., mixed) clauses in φ. Without loss of generality, assume that the clauses are numbered such that $C_1, \ldots, C_{m_N}$ are the mixed clauses. Moreover, we assume that φ is not satisfied by the all-zero assignment.

Define $k := \lceil (\alpha - 1) \cdot (m_N + n) \cdot f(n) \rceil$, where f denotes some unbounded, but arbitrarily slowly increasing function, e. g., the inverse of the Ackermann function.

From φ, we construct a Uniform Min-Ones-2SAT formula ψ in two steps. In the first step, we add new variables y_1, …, y_n and $z_{1,1}$, $z_{1,2}$, …, $z_{1,k}$, …, $z_{n,1}$, …, $z_{n,k}$ together with new clauses $(x_i \vee y_i)$, $(\overline{x_i} \vee \overline{y_i})$, $(y_i \vee z_{i,j})$, and $(\overline{y_i} \vee \overline{z_{i,j}})$, for all $i \in \{1, \ldots, n\}$ and all $j \in \{1, \ldots, k\}$.

In the second step, we add new variables u_1, …, u_{m_N} and v_1, …, v_{m_N} and replace the mixed clause $C_i = (x_{i_1} \vee \overline{x_{i_2}})$ by the four uniform clauses $(x_{i_1} \vee v_i)$, $(\overline{x_{i_2}} \vee \overline{v_i})$, $(v_i \vee u_i)$, and $(\overline{v_i} \vee \overline{u_i})$, for all $i \in \{1, \ldots, m_N\}$.

The satisfiability graph of the resulting formula ψ is shown for a sample formula in Figure 1.

Clearly, the formula ψ can be constructed from φ in polynomial time. We prove in the following that restricting an α-approximate assignment for ψ to the variables from X yields the desired asymptotic α-approximate solution for φ. For this, we prove the following claim:

> Let β be a satisfying assignment for φ setting c variables from X to one. Then every assignment γ for ψ which coincides with β on X satisfies ψ and has exactly $n + c \cdot k + m_N$ variables set to one. (1)

To prove this claim, we first consider the y- and z-variables. The respective clauses in ψ are constructed such that they formulate an exclusive-or constraint for x_i and y_i as well as an exclusive-or constraint for y_i and any of the $z_{i,j}$. Thus, if $\beta(x_i) = \gamma(x_i) = 1$, then $\gamma(y_i) = 0$ and thus $\gamma(z_{i_1}) = \ldots = \gamma(z_{i,k}) = 1$. If, on the other hand, $\beta(x_i) = \gamma(x_i) = 0$, then $\gamma(y_i) = 1$ and $\gamma(z_{i_1}) = \ldots = \gamma(z_{i,k}) = 0$. Overall, $c \cdot k$ of the z-variables and $n - c$ of the y-variables are set to one by γ.

We now consider the u- and v-variables. Let $C_i = (x_{i_1} \vee \overline{x_{i_2}})$ be any mixed clause in φ. The two clauses $(v_i \vee u_i)$ and $(\overline{v_i} \vee \overline{u_i})$ formulate an exclusive-or constraint for v_i and u_i. Thus, every satisfying assignment γ for ψ has to satisfy exactly m_N of these variables. It remains to show that it is possible to extend any satisfying assignment β for φ such that it also satisfies the clauses added in the second step of the construction. For this, we distinguish three cases according to how a mixed clause C_i is satisfied by β. At least one of the two literals has to be set to one by β. If $\beta(x_{i_1}) = 0$, then $\gamma(v_i) = 1$ is forced; if $\beta(x_{i_2}) = 1$, then this forces $\gamma(v_i) = 0$. If $\beta(x_{i_1}) = 1$ and $\beta(x_{i_2}) = 0$, then v_i can be set arbitrarily.

Overall, γ sets $c + c \cdot k + (n - c) + m_N = n + c \cdot k + m_N$ variables to 1 which proves the Claim (1).

We are now ready to analyze the approximation ratio achieved by this transformation. Let c_{opt} denote the minimum number of ones which have to be set to 1 in order to satisfy φ. Due to Claim (1), this implies that the minimum number of variables which have to be set to one for satisfying ψ is $n + c_{\text{opt}} \cdot k + m_N$. This means, an α-approximation algorithm for Uniform Min-Ones-2SAT computes an assignment γ_{approx} for ψ which sets at most $\alpha \cdot (n + c_{\text{opt}} \cdot k + m_N)$ variables to one. Let x denote the number of variables from X which are set to one by γ_{approx}. Following the argumentation above, the number x can also be counted as

$$x \leq \alpha \cdot (n + c_{\text{opt}} \cdot k + m_N) - m_N - x \cdot k - (n - x).$$

This leads to

$$x \leq \frac{1}{k} \cdot (\alpha \cdot (n + c_{\text{opt}} \cdot k + m_N) - m_N - n).$$

Thus, the approximation ratio of this approach can be estimated as

$$\begin{aligned} \frac{x}{c_{\text{opt}}} &\leq \frac{1}{c_{\text{opt}}} \cdot \frac{1}{k} \cdot (\alpha \cdot (n + c_{\text{opt}} \cdot k + m_N) - m_N - n) \\ &\leq \alpha + \frac{1}{k} \frac{1}{c_{\text{opt}}} \cdot (\alpha - 1)(n + m_N) \\ &\leq \alpha + \frac{1}{c_{\text{opt}} \cdot f(n)} \end{aligned}$$

which tends to α for n approaching infinity. □

5 An Exact Algorithm for Uniform Min-Ones-2SAT

The main goal of this section is to design a fast exact algorithm for Uniform Min-Ones-2SAT. In the whole section, we use the graph representation of this problem. Given the satisfiability graph $G_\varphi = (X, E_+ \cup E_-)$ of a uniform 2CNF formula φ, we are looking for a subset $S \subseteq X$ of vertices, such that $S \cap V(G_\varphi^+)$ is a minimum vertex cover on G_φ^+ and $S \cap V(G_\varphi^-)$ is an independent set on G_φ^-.

Our algorithm uses a standard technique know as branch-and-bound (see [10, 6]). The approach is based on the following observation about alternating trees in the satisfiability graph. Here, an *alternating tree* is a subtree T of the satisfiability

graph, rooted in some vertex r, such that all paths from the root alternate between positive and negative edges. An alternating tree is called *positive*, if all edges incident to its root r are positive, and it is called *negative*, if all edges incident to r are negative.

Observation 5 *Let $G_\varphi = (X, E_+ \cup E_-)$ be the satisfiability graph of a uniform 2CNF formula φ, and let x be some vertex of G_φ. If x gets assigned the value 1, this induces a value for every vertex of any negative alternating tree rooted in x. Moreover, if x gets assigned the value 0, this induces a value for every vertex of any positive alternating tree rooted in x.*

The overall strategy of our algorithm is to use some data reduction rules based on Observation 5 to eliminate all negative edges from the satisfiability graph and to subsequently apply a fast exact algorithm for the vertex cover problem on the remaining graph.

More precisely, the application of each rule can be divided into two steps. First, we guess the membership in the vertex cover for a constant number of vertices, and then, we deduce information about other vertices in the neighbourhood. Whenever this deduction process shows an inconsistency, the current branch of the search fails and is terminated immediately.

Once none of our reduction rules can be applied, all negative edges have been removed. Then the branching terminates and we employ an exact algorithm for the minimum vertex cover problem in order to obtain one of the final solutions. Out of all final solutions, we then pick the best one.

The data reduction rules have to be applied sequentially in the order described below. With each application of a rule, some subset of the vertices is chosen to be included either in the vertex cover C or in the non-vertex-cover set N. Whenever a vertex is assigned to be in C or in N, this is propagated along the corresponding alternating tree according to Observation 5. These rules are shown graphically in Figure 2.

1. Insert all vertices from $V(G_\varphi^-) \setminus V(G_\varphi^+)$ into N.
2. Use exhaustive search on all connected components containing at most four vertices to assign these vertices to C and N.
3. If there exists a triangle of unassigned vertices consisting of two positive edges $\{u, v\}$ and $\{v, w\}$ and one negative edge $\{u, w\}$, then v has to be inserted into C.
4. If there exists an unassigned vertex $v \in V(G_\varphi^+) \cap V(G_\varphi^-)$ that has at least two incident edges in E_+ and also two incident edges in E_-, consider two branches according to whether v is inserted into C or N.
5. If the graph contains an alternating cycle of length 4, branch over the two possible assignments for an arbitrary vertex on this cycle.
6. If the graph contains an alternating path on unassigned vertices that has length at least 4, branch over the two possible assignments for the third vertex of the path (if the length of path is exactly 4, this is the middle vertex).
7. If there are two unassigned vertices v and u connected by both a positive and a negative edge, exactly one of them has to belong to C. Consider two sub-cases:

(a) If all other edges incident to u are positive and all other edges incident to v are negative, insert u into C.
(b) Otherwise, branch over the two possible choices of inserting u or v into C.

8. If there exists an unassigned vertex v that is incident with two positive edges and one negative edge, branch over the two possibilities of inserting v into C or N.
9. If there exists a triangle of unassigned vertices consisting of one positive edge $\{u, v\}$ and two negative edges $\{u, w\}$ and $\{v, w\}$, branch over the choice of inserting u or v into C.
10. Insert all unassigned vertices incident to negative edges into N and their neighbours via positive edges into C and compute a vertex cover on the graph induced by the remaining unassigned vertices.

We now prove the correctness of this algorithm and analyze its running time. We start by describing the situation when none of the first nine rules can be applied any more.

Lemma 1. *Suppose that the first nine of the above data reduction rules can no further be applied. Let $G' = (V', E')$ be the graph induced by the still unassigned vertices, and let $E'_+ = E' \cap E_+$ and $E'_- = E' \cap E_-$ denote the set of positive and negative edges of G', respectively. Let $V'_+ \subseteq V'$ and $V'_- \subseteq V'$ be the set of vertices incident to some positive and negative edge in G', respectively.*

Then every vertex $x \in V'_-$ is incident to exactly one positive edge whose other endpoint is in V'_+.

Proof. We first show that, under the preconditions of the lemma, no two vertices in V'_- can be connected via a positive edge. Towards contradiction, suppose that there exist $x, y \in V'_-$ such that $\{x, y\} \in E'_+$. Note that this edge is included in a connected component of G' of size at least five due to rule 2. Since $x, y \in V'_-$, there exist $u, v \in V'_-$ such that $\{x, u\}, \{y, v\} \in E'_-$. Moreover, since rule 9 cannot be applied, we have $u \neq v$ and since also rule 7 cannot be applied, $x \neq v$ and $y \neq u$. Thus, all four vertices x, y, u, v have to be different.

We now consider the positive edge $\{u, p\}$ for $p \neq x$ that has to be present in the graph G' since otherwise rule 1 would be applicable. If vertex p would lie outside the set $\{x, y, u, v\}$, rule 6 would be applicable. Thus, $\{x, y, u, v, p\} = \{x, y, u, v\}$. If $p = v$, we would have an alternating cycle x, y, v, u, x of length 4, which would make rule 5 applicable. Thus, the only remaining possibility is $p = y$. But in this case, rule 3 would be applicable on the triangle x, y, u, x. Thus, the existence of the positive edge $\{x, y\}$ leads to a contradiction and we have proven that no two vertices in V'_- can be connected via a positive edge.

Now we prove the second claim of the lemma. As we have shown above, all positive edges incident to vertices from V'_- have to have their second endpoints in V'_+. The existence of a vertex v in V'_- that is an endpoint of two different positive edges would make rule 8 applicable, since v has to have a neighbour u via a negative edge, and u has to have a neighbor in V'_+ as well, due to rule 1. □

Theorem 2. *The branch-and-bound algorithm based on the rules 1 to 10 solves* Uniform Min-Ones-2SAT *in $\mathcal{O}(1.25993^n)$ time.*

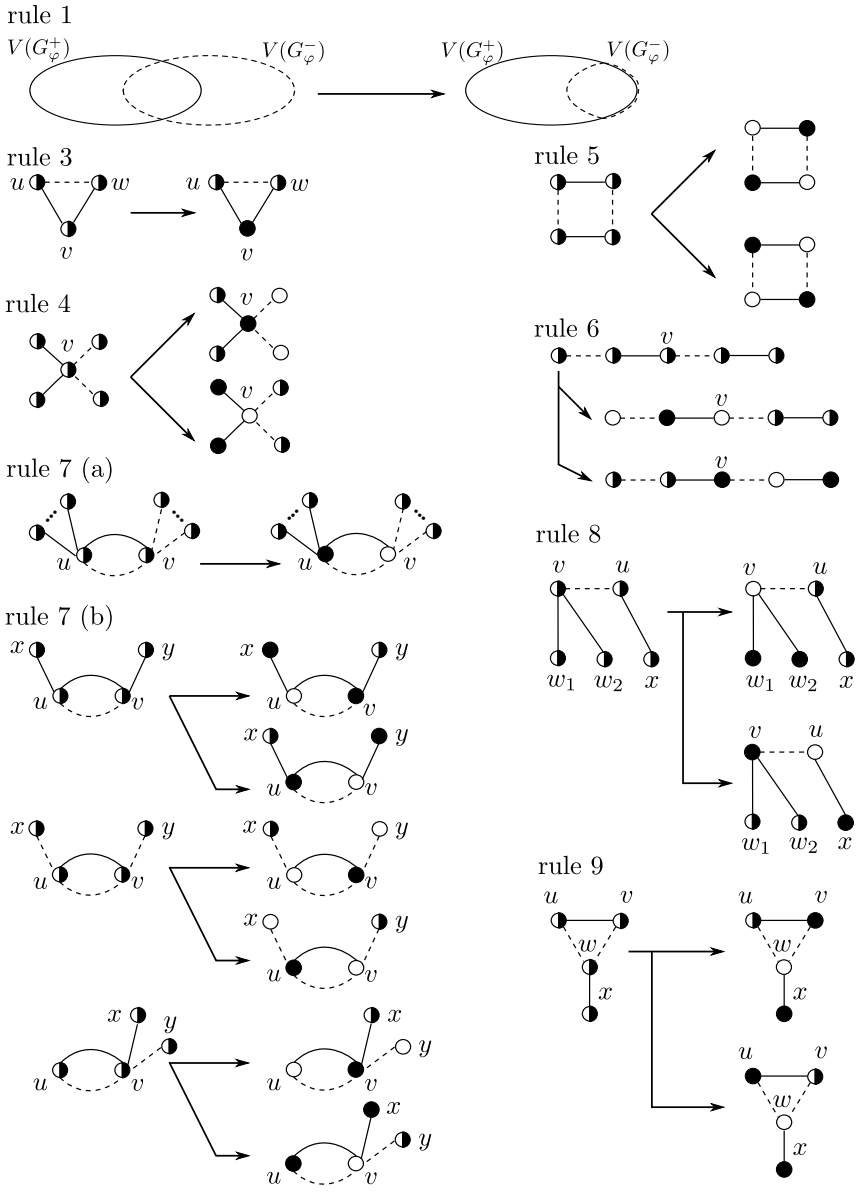

Fig. 2. An illustration of the data reduction rules of the presented exact algorithm for Uniform Min-Ones-2SAT. Solid lines denote positive edges, dashed lines denote negative edges. Vertices in C are drawn black, vertices in N are drawn white; the black-and-white vertices are still unassigned.

Proof. We first prove the correctness of the algorithm. We start with the non-branching rules. It is clear that a vertex that is incident to negative edges only, cannot appear in any minimum vertex cover on the positive edges. Thus, the application of rule 1 is correct. The correctness of rule 2 is obvious, since all connected components can be treated independently. For rule 3, assume that $v \in N$. Due to the two positive edges in the triangle, both u and w have to be in C, contradicting the negative edge between them. Thus, $v \in C$, and rule 3 is correct. Now consider rule 7 (a). The double edge between u and v implies that exactly one of these two vertices has to be included in C. Choosing $u \in C$ and $v \in N$ implies that all edges incident to u and v are satisfied. This leaves the highest possible degree of freedom for choosing the assignment for the neighbouring vertices and thus leads to a minimum vertex cover solution.

Considering the branching rules 4 to 6, 7 (b), and 8, we observe that the algorithm always branches over the two possible assignments for a single vertex and then applies Observation 5 for inferring the assignment for some further vertices. Therefore, the correctness of these rules follows directly from Observation 5. For the branching rule 9, it suffices to observe that at least one of the vertices u and v has to be included in C.

It remains to prove the correctness of rule 10. From Lemma 1, we know that every vertex x that is incident to a negative edge, is incident to exactly one positive edge, and furthermore, that the other endpoint y of this positive edge is incident to positive edges only. This means, instead of inserting x into C, we can insert y into C without changing the value of the solution. Thus, rule 10 and therefore, the whole algorithm are correct.

Now we analyze the time complexity of our algorithm. The main point here is that, in every branching step, we significantly reduce the number of unassigned vertices.

> Any application of one of the branching rules 4 to 6, 7 (b), 8, and 9 reduces the number of unassigned vertices in every branch by at least 3. (2)

In the following, we prove this separately for every branching rule.

For rule 4, assigning $v \in C$ in the first branch implies that its at least two neighbours via negative edges have to be included into N, giving a total of at least 3 newly assigned vertices. For assigning $v \in N$ in the second branch, analogously the at least two neighbours via positive edges have to be included into C.

For an alternating cycle of even length, it is easy to see that fixing the assignment for some arbitrary vertex triggers the assignment for all other vertices in the cycle. Thus, the application of rule 5 reduces the number of unassigned vertices by at least 4.

For an alternating path containing at least 4 edges, the third vertex v of this path is the root of two different alternating paths of length at least 2, one starting with a positive edge, the other starting with a negative edge. Due to Observation 5, this implies that any assignment for v triggers the assignment for one of these alternating paths, and thus for at least two additional vertices. Thus, also the

branching according to rule 6 reduces the number of unassigned vertices by at least 3.

When applying rule 7 (b), we consider the case of a double edge $\{u, v\}$ and branch over the two possible assignments for it. Additionally, we know that rule 7 (a) is not applicable. Thus, we distinguish three cases for the analysis. Assume first that u and v are incident to some additional positive edge each. Let x and y be their neighbours via these edges, respectively. Then, in the branch where u is included in N, its neighbour x has to be inserted into C. Analogously, in the other branch, y has to be inserted into C.[3] Now assume that u has a neighbour x via a negative edge and v has a neighbour y via a negative edge. Analogously as in the preceding case, if u is included in C, then x has to be in N, and if v is included in C, then y has to be in N.[4] The remaining case is that v (or u) is incident to a positive and a negative edge with endpoints x and y, respectively. In this case, the assignment for v triggers the assignment of either x or y according to Observation 5. Thus, applying rule 7 (b) assigns values to at least three vertices in any case.

In the situation of rule 8, denote by u the neighbour of v via the negative edge and by w_1 and w_2 the two neighbours of v via the positive edges. Due to rule 1, there exists a further vertex x which is adjacent to u via a positive edge. Since rule 3 is not applicable, $x \neq w_1$ and $x \neq w_2$. According to Observation 5, w_1 and w_2 have to be included in C in the branch where v is inserted into N. On the other hand, letting $v \in C$ triggers the assignment for u and x.

Finally, we consider rule 9. Due to rule 1, there exists another vertex x which is adjacent to w via a positive edge. Since rule 7 is not applicable, we know that $x \neq u$ and $x \neq v$. Independent of whether u or v is inserted into C, we know that $w \in N$ and thus $x \in C$.

We have now proved that every branching step reduces the number of unassigned vertices by at least 3.

Claim (2) leads to the recurrence

$$T(n) = 2 \cdot T(n-3)$$

for the time complexity of our approach for inputs with n vertices. Applying standard techniques for solving recurrence equations yields an upper bound of

$$T(n) \leq \left(\sqrt[3]{2}\right)^n \leq 1.25993^n.$$

In addition, we have to consider the time needed for the vertex cover computation after the application of rule 10. For this, we can use any exact vertex cover algorithm. Since every minimum vertex cover in a graph is a maximum independent set in the complement graph (i. e., in the graph resulting from swapping edges and non-edges), we can also use any exact algorithm for the maximum independent set problem on the complement graph. One algorithm for this is due to Robson [12] and runs in $\mathcal{O}(1.211^n)$ time. Thus, the overall running time of our algorithm can be bounded by $\mathcal{O}(1.25993^n)$. □

[3] Note that the case where $x = y$ is already covered by rule 3.

[4] Note that in the case $x = y$ no branching is necessary since this vertex has to be in N, anyway.

6 Conclusion

In this paper, we introduced the Uniform Min-Ones-2SAT problem as a means for modelling a problem of haplotype classification. We analyzed its approximability and presented a moderately exponential-time exact algorithm for it. While we have seen that Uniform Min-Ones-2SAT and general Min-Ones-2SAT are equally hard with respect to approximability, it remains as an interesting open problem to extend the exact algorithm to the general case.

Moreover, from the viewpoint of the biological application, it would be very interesting to generalize the classification model to include all types of individuals as listed in Table 1 or to allow for some errors in the data.

References

1. Bengt Aspvall, Michael F. Plass, and Robert Endre Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, 8(3):121–123, 1979.
2. Hans-Joachim Böckenhauer and Dirk Bongartz. *Algorithmic Aspects of Bioinformatics*. Natural Computing Series. Springer-Verlag, 2007.
3. Paola Bonizzoni, Gianluca Della Vedova, Riccardo Dondi, and Jing Li. The haplotyping problem: An overview of computational models and solutions. *Journal of Computer Science and Technology*, 18(6):675–688, 2003.
4. Irit Dinur and Samuel Safra. On the hardness of approximating minimum vertex cover. *Annals of Mathematics*, 162:439–485, 2005.
5. Dan Gusfield and Leonard Pitt. A bounded approximation for the minimum cost 2-sat problem. *Algorithmica*, 8(2):103–117, 1992.
6. Juraj Hromkovič. *Algorithmics for Hard Problems. Introduction to Combinatorial Optimization, Randomization, Approximation, and Heuristics*. Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin, 2003.
7. George Karakostas. A better approximation ratio for the vertex cover problem. Technical Report TR04-084, ECCC, 2004.
8. Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within 2-epsilon. *Journal of Computer and System Sciences*, 74(3):335–349, 2008.
9. Jun Kiniwa. Approximation of self-stabilizing vertex cover less than 2. In *Self Stabilizing Systems 2005*, pages 171–182, 2005.
10. E. L. Lawler and D. E. Wood. Branch-and-bound methods: A survey. *Operations Research*, 14(4):699–719, 1966.
11. Jing Li and Tao Jiang. A survey on haplotyping algorithms for tightly linked markers. *Journal of Bioinformatics and Computational Biology*, 6(1):241–259, 2008.
12. J. M. Robson. Algorithms for maximum independent sets. *Journal of Algorithms*, 7(3):425–440, 1986.