# Absorbing Random Walks and the NAE2SAT Problem

K. Subramani*

LDCSEE,
West Virginia University,
Morgantown, WV
ksmani@csee.wvu.edu

**Abstract.** In this paper, we propose a simple, randomized algorithm for the NAE2SAT problem; the analysis of the algorithm uses the theory of symmetric, absorbing random walks. NAESAT (Not-All-Equal SAT) is the variant of the Satisfiability problem (SAT), in which we are interested in an assignment that satisfies all the clauses, but falsifies at least one literal in each clause. We show that the NAE2SAT problem admits an extremely simple literal-flipping algorithm, in precisely the same way that 2SAT does. On a satisfiable instance involving $n$ variables, our algorithm finds a satisfying assignment using at most $\frac{9}{4}n^2$ verification calls with probability at least $\frac{5}{6}$. The randomized algorithm takes $O(1)$ extra space, in the presence of a verifier and provides an interesting insight into checking whether a graph is bipartite. It must be noted that the bounds we derive are much sharper than the ones in [1].

## 1 Introduction

This paper details a randomized algorithm for the problem of determining whether an instance of NAE2SAT is satisfiable. To recapitulate, NAESAT is the version of clausal satisfiability (SAT), in which we seek an assignment that satisfies all the clauses, while falsifying at least one literal in each clause. NAESAT is known to be NP-complete, even when there are at most three literals per clause (NAE3SAT) [2]. The NAE2SAT problem is the variant of NAESAT in which there are exactly two literals per clause and is known to be solvable in polynomial time. For instance, one could use each clause as a partitioning constraint and decide an instance in linear time. From a complexity-theoretic perspective, [3] established that NAE2SAT is in the complexity class SL; more recently, [4] proved that SL=L, from which it follows that NAE2SAT is in L. Our algorithm is based on the literal-flipping algorithm proposed in [1] for the 2SAT problem and is extremely space-efficient. In particular, on a satisfiable NAE2SAT instance having $n$ variables and $m$ clauses, the expected number of literal-flips to find the satisfying assignment is $\frac{n^2}{4}$. Further, if a satisfying assignment is not found within $\frac{9}{4}n^2$ literal-flips, then the probability that the instance is not satisfiable is at least $\frac{5}{6}$.

The principal contributions of this paper are as follows:

(a) The design and analysis of a randomized, literal-flipping algorithm for the NAE2SAT problem.

* This research was supported in part by a research grant from the Air-Force Office of Scientific Research under contract FA9550-06-1-0050.

F.P. Preparata, X. Wu, and J. Yin (Eds.): FAW 2008, LNCS 5059, pp. 89–100, 2008.

(b) Some new observations on the convergence times of absorbing random walks.
(c) Establishing the complexity of NAE2SATPOS, which is the variant of NAE2SAT, in which all literals are positive.

## 2 Preliminaries

Let $\phi = C_1 \wedge C_2 \ldots \wedge C_m$ denote a 2CNF formula on the literal set $L = \{x_1, \bar{x_1}, x_2, \bar{x_2} \ldots x_n, \bar{x_n}\}$.

**Definition 1.** *The Not-All-Equal satisfiability problem for 2CNF formulas (NAE2SAT), is concerned with checking whether there exists a satisfying assignment to $\phi$, such that at least one literal in each clause is set to* **false**. *If such an assignment exists, then $\phi$ is said to be nae-satisfiable.*

For instance, the 2CNF formula $\phi = (x_1, x_2) \wedge (x_2, x_3)$ is nae-satisfiable ($x_1 = \mathbf{true}$ $x_2 = \mathbf{false}$ $x_3 = \mathbf{true}$), while the 2CNF formula $\phi = (\bar{x_1}, x_2) \wedge (x_1, x_2))$ is not nae-satisfiable.

Without loss of generality, we assume that each clause has *exactly* two literals, since $\phi$ cannot be nae-satisfiable, if it has a clause with exactly one literal.

In order to understand our approach for the NAE2SAT problem, we discuss a graph theoretic approach. Given a 2CNF formula $\phi(x)$, we can construct its nae-graph $G_\phi(x)$ as follows:

(a) For each literal $x_i$, create a vertex labeled $x_i$. Note that corresponding to boolean variable $x_i$, there are two literals, viz., $x_i$, and its complement literal, $\bar{x_i}$.
(b) Corresponding to each clause $(x_i,\ y_j)$ (say), add the undirected edges $\bar{y_j} - x_i$ and $\bar{x_i} - y_j$.

The graph that is created is similar but not identical to the implication graph technique for 2SAT discussed in [5]. In particular, note that in our case, the graph edges are undirected, whereas in [5], the edges represent implications and are therefore directed. The implication graph $G$ models the fact that each 2-literal clause $(x_i,\ \bar{y_j})$ can be thought of as a pair of implications $\bar{x_i} \rightarrow y_j$ and $\bar{y_j} \rightarrow x_i$ connected conjunctively. In [5], it is shown that the $\phi(x)$ is unsatisfiable if and only there exists a path from a vertex $x_i$ to its complement vertex $\bar{x_i}$ and vice versa. We now provide a test for the nae-satisfiability of $\phi(x)$.

**Theorem 1.** *Let $\phi(x)$ denote a CNF formula and $G_{\phi(x)}$ denote the corresponding nae-graph built as per the above rules. Then $\phi(x)$ is nae-unsatisfiable if and only there exists a path from a vertex $x_i$ to its complement vertex $\bar{x_i}$ in $G_{\phi(x)}$.*

**Proof:** Assume that there exists a path from a vertex $x_i$ to its complement vertex $\bar{x_i}$ denoted by $x_i \rightarrow x_1 \rightarrow \ldots \rightarrow x_k \rightarrow \bar{x_1}$. As per the construction of the implication graph, the following clauses are part of the clause set $\phi(x)$: $(\bar{x_i}, x_1)(\bar{x_1}, x_2) \ldots (\bar{x_k}, \bar{x_i})$. Now consider an assignment in which $x_i$ is set to **true**. It follows that $x_1$ must be set to **true**. Arguing similarly, $x_2, x_3, \ldots, x_k$ must all be set to **true**. However, the last clause $(\bar{x_k}, \bar{x_i})$ has both literals set to **false** and is therefore not nae-satisfied. Likewise,

consider an assignment in which $x_i$ is set to **false**. This forces $x_1$ to be **false**; arguing similarly, $x_2, x_3, \ldots, x_k$ must all be set to **false**. However, this forces both literals in the last clause to be **true**, thereby falsifying it from a nae-sat perspective.

The converse can be argued similarly, albeit with some effort involving case-based analysis. □

The above theorem immediately establishes that NAE2SAT can be solved in linear time using a variant of the connected components approach discussed in [5]. All that we need to ensure that is that no literal is reachable from its complement.

### 2.1 Assignment Verification for NAE2SAT

Consider an instance $F$ of NAE2SAT, i.e., a 2CNF formula on the $x$ variables only. Let $\vec{\mathbf{a}}$ represent a $\{\mathbf{true}, \mathbf{false}\}$ assignment to the variables of $F$. It is straightforward to verify whether $\vec{\mathbf{a}}$ nae-satisfies $F$, by substituting the value of literals in each clause, as specified by $\vec{\mathbf{a}}$ and checking that all clauses evaluate to **true**, with at least one literal set to **false** in each clause. We now present an implication graph interpretation of a nae-satisfying assignment. Let $G$ represent the implication graph of $F$ and consider the labeling of the vertices of $G$, as per $\vec{\mathbf{a}}$.

**Theorem 2.** *An assignment $\vec{\mathbf{a}}$ nae-satisfies a 2CNF formula $\phi(x)$, if and only if there is no arc, with tail vertex assigned* **true** *and head vertex assigned to* **false***, i.e., a (***true**→**false***) arc* **or** *a (***false**→**true***) arc, in the implication graph $G_\phi(x)$ of $\phi(x)$.*

**Proof:** Let $(x_i,\ x_j)$ denote an arbitrary clause of $\phi(x)$; this clause contributes the two arcs $l_1 : \bar{x_i} \rightarrow x_j$ and $l_2 : \bar{x_j} \rightarrow x_i$ to the implication graph $G$. If this clause is satisfied by the assignment $\vec{\mathbf{a}}$, either $x_i$ is set to **true** and $x_j$ is set to **false** or $x_j$ is set to **true** and $x_i$ is set to **false**. If $x_i$ is set to **true** and $x_j$ is set to **false**, the tail of arc $l_1$ is set to **false** and so is its head, while the head of arc $l_2$ is set to **true** and so is its head. We thus see that there is no (**true**→**false**) arc or (**false**→**true**) arc. The case when $x_j$ is assigned to **true** and $x_i$ is assigned to **false** can be argued symmetrically.

Now consider the case in which both $x_i$ and $x_j$ are set to **false** in $\vec{\mathbf{a}}$. Both the arcs $l_1$ and $l_2$ are (**true**→**false**) arcs. Similarly, the case in which $x_i$ and $x_j$ are both set to **true**, results in $l_1$ and $l_2$ both becoming (**false**→**true**) arcs. □

From Theorem 2, it follows that given an assignment to a NAE2SAT instance $\phi(x)$, the verification process consists of scanning through the implication graph $G_\phi(x)$ and ensuring that there does not exist a (**true**→**false**) or (**false**→**true**) arc. If the instance $\phi(x)$ has $m$ clauses and $n$ variables, then as per the implication graph construction, $G_\phi(x)$ has $2 \cdot m$ arcs. Hence, the verification process can be accomplished in $O(m)$ time.

### 2.2 Complexity Classes

**Definition 2.** *A language $\mathcal{L}$ is said to be in* RL*, if there exists a randomized algorithm which recognizes it in logarithmic space.*

**Definition 3.** *The undirected $s-t$ connectivity problem (USTCON) is defined as follows: Given an undirected graph $G = \langle V, E \rangle$ and two vertices, $s, t \in V$, is $t$ reachable from $s$?*

**Definition 4.** *A language $\mathcal{L}$ is said to be in* SL*, if it can be log-space reduced to USTCON.*

## 3 Markov Chains and Random Walks

This section contains preliminaries on Markov Chains and Random Walks that may not be familiar to non-experts in Probability Theory. I have included this material so that the paper as a whole is accessible to researchers in both Computer Science and Statistics. If the reviewers feel that this section does not add value to the paper, it will be removed.

There are two types of theorems that are described here; the first type are known results for which appropriate references are provided and the second type are results which are proved here for the first time, to the best of our knowledge. The latter theorems are marked as **O** for original.

Let $\mathcal{R}$ denote a stochastic process which can assume one of the following values: $S = \{0, 1, 2, \ldots n\}$. If $\mathcal{R}$ assumes the value $i$ at a particular time, then it is said to be in state $i$ at that time. Further, suppose that when it is in state $i$, the probability that it will move to state $j$, at the next instant is a fixed constant $p_{ij}$, independent of the history of the process. $\mathcal{R}$ is said to be a *Finite Markov Chain* over the state space $S$.

**Definition 5.** *A Finite Markov Chain over the state space $S = \{0, 1, \ldots, n\}$ is said to be a* Random Walk*, if for some fixed constants $p \in (0, 1)$, $p_0 \in [0, 1]$, $p_n \in [0, 1]$,*

$$\begin{aligned} p_{i\,i+1} = p = 1 - p_{i\,i-1},\ i = 1, 2, \ldots, n-1, \\ p_{0\,1} = p_0 \\ p_{n\,n-1} = p_n \end{aligned}$$

**Definition 6.** *A random walk $\mathcal{R}$ is said to be absorbing, if $p_{0\,1} = p_{n,n-1} = 0$.*

States $0$ and $n$ are said to be absorbing barriers of the random walk.

**Definition 7.** *An absorbing random walk is said to be symmetric, if $p = \frac{1}{2}$.*

The rest of this paper will be concerned with absorbing, symmetric random walks (ASR walks) only. An ASR walk has the following game-theoretic interpretation: Imagine a drunkard on a straight road, with his house on one end and a bar on the other. Assume that the drunkard is at some point $i$, in between the house and the bar. Point $i$ is the initial position of the game. In each state the drunkard is currently in, he takes one step towards the bar with probability one-half or one step towards his house with probability one-half. The game is over when the drunkard reaches either his home or the bar.

**Definition 8.** *The absorption time of state $i$, $0 \leq i \leq n$, in an ASR walk $\mathcal{R}$, is the expected number of steps for $\mathcal{R}$ to reach an absorbing state, given that it is currently in state $i$.*

The absorption time of state $i$, is denoted by $t(i)$; we thus have,

$$t(i) = \mathbf{E}[\text{ASR walk } \mathcal{R} \text{ to reach 0 or n} \mid \mathcal{R} \text{ is currently in state i}]$$

**Definition 9.** *The absorption time of an ASR walk $\mathcal{R}$ is defined as the maximum absorption time over all the states in $\mathcal{R}$.*

The absorption time of $\mathcal{R}$ is denoted by $\mathcal{R}_t$; we thus have,

$$\mathcal{R}_t = \max_{0 \le i \le n} t(i)$$

The literature has shown that every state in an absorbing, symmetric random walk is recurrent, i.e., the probability that a state is ever reached is 1 [6]. From the perspective of algorithmic efficiency, mere recurrence is not sufficient; it is important that the absorption time of a state is s a small number, preferably a polynomial function of the total number of states.

We now proceed to compute the absorption time of various states in an ASR walk $\mathcal{R}$; in order to carry out this computation, we need the following technical lemma that helps us to compute the expectation of a random variable by conditioning it on a different random variable. This lemma has been proved in [7].

**Lemma 1.** *Let $X$ and $Y$ denote two random variables; let $\mathbf{E}[X \mid Y]$ denote that function of the random variable $Y$, whose value at $Y = y$ is $\mathbf{E}[X \mid Y = y]$. Then,*

$$\mathbf{E}[X] = \mathbf{E}[\mathbf{E}[X \mid Y]]. \tag{1}$$

*In other words,*

$$\mathbf{E}[X] = \sum_{y} \mathbf{E}[X \mid Y = y] \cdot \mathbf{Pr}[Y = y]. \tag{2}$$

We use $t(i)$ to denote the absorption time of state $i$ and $\mathcal{R}_t$ to denote the absorption time of the ASR walk $\mathcal{R}$. Observe that if the ASR walk is currently in state $i$, where $i$ is a non-absorbing state, then at the next step, it will be in state $(i+1)$ with probability one-half and it will be in state $(i-1)$, with probability one-half. In the former case, the absorption time for state $i$ is $(1 + t(i+1))$, while in the latter case, it is $(1 + t(i-1))$. Noting that $t(0) = t(n) = 0$, we apply Lemma (1) to derive the following set of equations for computing the absorption times of states in the ASR walk.

$$\begin{aligned} t(0) &= 0 \\ t(i) &= \frac{1}{2} \cdot (t(i-1) + 1) + \frac{1}{2} \cdot (t(i+1) + 1),\ 0 < i < n \\ t(n) &= 0 \end{aligned} \tag{3}$$

Note that System (3) contains $(n+1)$ equations and $(n+1)$ unknowns and can be represented as: $\mathbf{A} \cdot \vec{\mathbf{x}} = \vec{\mathbf{b}}$, where, $\vec{\mathbf{x}}$ is an $(n+1)$-vector representing the expected absorption

times of the states ($[t(0),\ t(1),\ \ldots, t(n)]^T$), $\vec{\mathbf{b}}$ is an $(n+1)$-vector $[0,\ 1,\ 1,\ \ldots, 1,\ 0]^T$, and $\mathbf{A}$ is the $(n+1) \times (n+1)$ matrix represented by:

$$\begin{bmatrix} 1 & 0 & \ldots & 0 \\ -\frac{1}{2} & 1 & -\frac{1}{2} & \ldots \\ \vdots & \vdots & \vdots & \vdots \\ \ldots & -\frac{1}{2} & 1 & -\frac{1}{2} \\ 0 & 0 & \ldots & 1 \end{bmatrix}$$

Row $i$ $(0 < i < n)$, of $\mathbf{A}$ has 1 in the diagonal entry and a negative one-half in the entry preceding and succeeding it. All other entries in this row are zero. Row 0 and Row $n$ are unit vectors with the unit entry occupying the diagonal entry. It is not hard to see that $\mathbf{A}$ is non-singular and therefore, System (3) has a unique solution.

We need the following lemma to aid us in solving System (3).

**Lemma 2**

$$t(i) = t(n-i),\ \forall i = 0, 1,\ \ldots, n$$

**Proof:** By symmetry of the random walk. □

The technique to solve System (3) (which is a non-trivial contribution) appears in the journal version of the paper. We show that in the worst case, the absorption time of a state (i.e., the expected number of steps for the random walk to reach an absorbing state, from a state) in an ASR walk is $\frac{n^2}{4}$, i.e., $\max_{0 \leq i \leq n} t(i) \leq \frac{n^2}{4}$.

**Theorem 3.** *Let* $\mathbf{Var}[t(i)]$ *denote the variance of the absorption time of state* $i$*, in the ASR walk* $\mathcal{R}$*. Then,*

$$\max_{0 \leq i \leq n} \mathbf{Var}[t(i)] \leq \frac{2}{3} n^4.$$

**Proof:** Will provide in revised version. □

The following theorem is known as Chebyshev's inequality and is proved in [7] among other places.

**Theorem 4.** *Let* $X$ *denote a random variable, with mean* $\mathbf{E}[X]$ *and variance* $\sigma^2$*. Then, for any* $k > 0$*, we have,*

$$\mathbf{Pr}[|X - \mathbf{E}[X]| \geq k] \leq \frac{\sigma^2}{k^2}$$

**Theorem 5.** *(***O***) Given an ASR walk* $\mathcal{R}$*, which is initially in an arbitrary state* $i$*,* $0 \leq i \leq n$*, the probability that* $\mathcal{R}$ *does not reach an absorbing state in* $\frac{9}{4}n^2$ *steps is at most* $\frac{1}{6}$*.*

**Proof:** Let $X$ denote the worst-case number of steps taken by the ASR walk $\mathcal{R}$ to reach an absorbing state. We are interested in the quantity $\mathbf{Pr}[X \geq \frac{9}{4}n^2]$.

Note that,

$$\begin{aligned}
\mathbf{Pr}[X \geq \frac{9}{4}n^2] &= \mathbf{Pr}[X - \frac{n^2}{4} \geq 2 \cdot n^2] \\
&\leq \mathbf{Pr}[|X - \frac{n^2}{4}| \geq 2 \cdot n^2] \\
&= \mathbf{Pr}[X - \mathbf{E}[X]| \geq 2 \cdot n^2] \\
&\leq \frac{\frac{2}{3}n^4}{(2n^2)^2}, \text{ using Chebyshev's inequality} \\
&= \frac{1}{6}
\end{aligned}$$

□

### 3.1 The Convergence Numbers

Let $G(n, k)$ denote the expected number of steps taken by an ASR walk to reach an absorbing state, assuming that it is currently in state $k$, $0 \leq k \leq n$, i.e., $G(n, k)$ is the absorption time of state $k$.

Table (1) represents the computed values of $G(n, k)$ for small values of $n$. For each value of $n$, the corresponding row stores the absorption time for all $n+1$ initial positions of the random walk.

**Table 1.** Convergence Numbers

| Number of variables | Starting Point of Walk |
|---|---|
| $n = 0$ | 0 |
| $n = 1$ | 0 0 |
| $n = 2$ | 0 1 0 |
| $n = 3$ | 0 2 2 0 |
| $n = 4$ | 0 3 4 3 0 |
| $n = 5$ | 0 4 6 6 4 0 |
| $n = 6$ | 0 5 8 9 8 5 0 |
| $n = 7$ | 0 6 10 12 12 10 6 0 |

Each of the following theorems (which do not appear in the literature, to the best of our knowledge) can be proved using induction.

**Theorem 6.** *For all* $n \geq 2$, $G(n, 1) = G(n-1, 1) + 1$.

**Theorem 7.** *For all* $n \geq 3$, $G(n, 2) = G(n-2, 2) + 2$.

**Theorem 8.** *For all* $n \geq k$, $G(n, k) = G(n-k, k) + k$.

**Theorem 9.** *For all* $n$, $G(n, k) = G(n, n-k)$.

## 4 Algorithm and Analysis

Algorithm 4.1 is our strategy to solve the NAE2SAT problem.

**Algorithm 4.1.** Randomized algorithm for the NAE2SAT problem

**Function** NAE2SAT-SOLVE($G_\phi(x)$)
1: {The 2CNF formula $\phi(x)$ is input through its implication graph $G_\phi(x)$.}
2: {We assume that $\phi(x)$ has $n$ variables and $m$ clauses, so $G_\phi(x)$ has $2 \cdot n$ vertices and $2 \cdot m$ arcs.}
3: Let $T$ be an arbitrary truth assignment to the variables of $\phi(x)$.
4: Update $G_\phi(x)$ with $T$.
5: {We say that an arc in $G_{\phi(x)}$ is broken, if under the current assignment it is a **true**→**false** arc or a **false**→**true** arc.}
6: $count = 0$;
7: **if** (there does not exist a broken arc in $G_\phi(x)$) **then**
8: **return** ("$\phi(x)$ is nae-satisfiable")
9: **end if**
10: **while** (there exists at least one broken arc in $G_\phi(x)$) **and** ($count \leq \frac{9}{4}n^2$) **do**
11: Select any broken arc in $G_\phi(x)$, say $x_1 \rightarrow x_2$.
12: Flip a fair coin to pick one of $x_1$ and $x_2$.
13: **if** ($x_1$ is selected) **then**
14: Flip $x_1$; i.e., complement its assignment.
15: **else**
16: Flip $x_2$.
17: **end if**
18: Adjust $T$ and $G_\phi(x)$ to reflect the changed assignments.
19: **if** ($T$ now satisfies $\phi(x)$. i.e., there is no broken arc in $G_\phi(x)$) **then**
20: **return** ("$\phi(x)$ is nae-satisfiable.")
21: **else**
22: $count = count + 1$.
23: **end if**
24: **end while**
25: **return** ("$\phi(x)$ is probably nae-unsatisfiable.")

### 4.1 Analysis

Observe that if Algorithm 4.1 claims that $\phi(x)$ is nae-satisfiable, then $\phi(x)$ definitely has a nae-satisfying assignment, i.e, the assignment $T$, which causes the algorithm to terminate. On the other hand, if Algorithm 4.1 claims that $\phi(x)$ is not nae-satisfiable, then it is possible that $\phi(x)$ is still nae-satisfiable; we now show that the probability of this occurrence is less than $\frac{1}{6}$.

**Lemma 3.** *Let $\phi(x)$ denote a 2CNF formula; if assignment $\mathbf{x}$ nae-satisfies $\phi(x)$, then so does assignment $\mathbf{x^c}$, where $\mathbf{x^c}$ is derived from $\mathbf{x}$, by complementing the assignment to each variable in $\mathbf{x}$. The tuple $(\mathbf{x}, \mathbf{x^c})$ is called a complementary pair.*

**Proof:** Since, $\mathbf{x}$ nae-satisfies $\phi(x)$, it sets one literal to **true** and one literal to **false** in each clause. Under $\mathbf{x^c}$, the literals which are set to **true** become **false** and vice versa. It follows that each clause is still nae-satisfied and therefore, so is $\phi(x)$. □

Assume that $\phi(x)$ is nae-satisfiable and let us focus on a particular nae-satisfying complementary pair of assignments $\hat{T}$ and $\hat{T}^c$. Let $T$ denote the current assignment to the variables of $\phi(x)$. If $T$ is a nae-satisfying assignment, we are done. If it is not, then there is a clause, say $(x_i, x_j)$ that is not nae-satisfied by $T$. There are two cases to consider:

(i) Both $x_i$ and $x_j$ are set to **false** in $T$ - In $\hat{T}$, at least one of these two literals is set to **true**; likewise, in $\hat{T}^c$, at least one of these literals is set to **false**. Hence choosing one of them uniformly and at random, moves $T$ closer to $\hat{T}$, with probability at least one-half and closer to $\hat{T}^c$ with probability one-half. In other words, after the literal flip, with probability one-half, $T$ agrees with $\hat{T}$ in one more variable and with probability one-half, $T$ agrees with $\hat{T}^c$ in one more variable.

(ii) Both $x_i$ and $x_j$ are set to **true** in $T$ - In $\hat{T}$, at least one of the two literals is set to **false** and in $\hat{T}^c$, at least one of the two literals is set to **true**. Hence choosing one of the two literals uniformly and at random and flipping it, moves $T$ closer (by one variable) to $\hat{T}$ with probability at least one-half and closer to $\hat{T}^c$ (by one variable) with probability at least one-half.

Let $r(i)$ denote the expected number of literal-flips for Algorithm 4.1 to take the current assignment $T$ to the nae-satisfying assignment $\hat{T}$ or its complement $\hat{T}^c$, assuming that $T$ differs from $\hat{T}$ on exactly $i$ variables. By our previous arguments, it is clear that $T$ differs $\hat{T}^c$ in exactly $(n-i)$ variables. Note that $r(0) = 0$, since if the current assignment differs from $\hat{T}$ on 0 variables, then it is a nae-satisfying assignment. Likewise, $r(n) = 0$, since if the current assignment differs from $\hat{T}$ on all $n$ variables, then it must differ from $\hat{T}^c$ on exactly 0 variables, i.e., $T$ must coincide with the nae-satisfying assignment $\hat{T}^c$.

Applying Lemma (1) to the discussion above, for each $i$, $0 < i < n$, we must have

$$\begin{aligned} r(i) &= \frac{1}{2}\left(r(i-1)+1\right) + \frac{1}{2}(r(i+1)+1) \\ &= \frac{1}{2}r(i-1) + \frac{1}{2}r(i+1) + 1 \end{aligned} \tag{4}$$

But System (4) in conjunction with the boundary conditions is precisely the defining system of an ASR walk $\mathcal{R}$. We therefore conclude that:

**Theorem 10**

$$\max_{0\le i\le n} r(i) \le \frac{n^2}{4}.$$

$$\max_{0\le i\le n} \mathbf{Var}[r(i)] = \frac{2}{3}n^4.$$

**Corollary 1.** *Algorithm 4.1 is a Monte-Carlo algorithm for the NAE2SAT problem; on a nae-satisfiable instance, the probability that it does not find a nae-satisfying assignment is at most* $\frac{1}{6}$.

## 5 Graph Bipartiteness

In this section, we apply the techniques of Algorithm 4.1 to derive a Monte Carlo algorithm for the problems of checking whether an undirected graph is 2-colorable. Without loss of generality, we assume that the vertices of the graph need to be colored from the set {**red, blue**}. A particular coloring **c** for the vertices of $G$, is inconsistent if there exists at least one edge $e$ such that both its endpoints are colored **blue** or **red**. If no such edge exists, **c** is said to be a consistent (valid) coloring.

**Algorithm 5.1.** Randomized algorithm for the Undirected Graph 2-coloring problem.

**Function** GRAPH-2-COLOR($G$)
1: {We assume that $G$ has $n$ variables and $m$ edges.}
2: Let **c** be an arbitrary color assignment of **red** and **blue** to the vertices of $G$.
3: $count = 0$;
4: **if** (**c** is a valid coloring) **then**
5: **return** ("$G$ is 2-colorable.")
6: **end if**
7: **while** ( (**c** is an inconsistent coloring)**and** ($count \leq \frac{9}{4} \cdot n^2$)) **do**
8: Pick an edge $e = (x_i, x_j) \in E$ such that $c[x_i] = c[x_j] =$ **red** or $c[x_i] = c[x_j] =$ **blue**
9: Flip a fair coin to pick one of $x_i$ and $x_j$.
10: **if** ($x_i$ is selected) **then**
11: Change its color to **blue**, if it was **red** and to **red**, if it was **blue**.
12: **else**
13: Change the color of $x_j$ to **blue**, if it was **red** and to **red**, if it was **blue**.
14: **end if**
15: Update **c** accordingly.
16: **if** (the current color assignment is valid) **then**
17: "$G$ is 2-colorable")
18: **else**
19: $count = count + 1$.
20: **end if**
21: **end while**
22: **return** ("$G$ is probably not 2-colorable.)

Algorithm 5.1 is a Monte Carlo algorithm for checking whether a graph is bipartite.

### 5.1 Analysis

The graph bipartiteness problem shares an important property with the NAE2SAT problem, in that if a particular coloring **c** is a valid 2-coloring of a graph, then so is its complement coloring $\bar{\mathbf{c}}$. Clearly if Algorithm 5.1 returns "yes", then the input instance $G$ is bipartite. However, if the algorithm claims that $G$ is not 2-colorable, then it could be incorrect. The error bound analysis is identical to the one for NAE2SAT, since at each step, Algorithm 5.1 moves one step closer to a valid coloring or one step closer to the complement of that valid coloring. Algorithm 5.1 can therefore also be modeled as

a one dimensional random walk with one reflecting barrier and one absorbing barrier. Accordingly, we get,

**Theorem 11.** *Algorithm 5.1 is a Monte Carlo algorithm for the problem of checking whether an undirected graph is 2-colorable. If the input graph is not 2-colorable, the algorithm always returns the correct answer; if the input graph is 2-colorable, the probability that the algorithm returns the incorrect answer is at most* $\frac{1}{6}$.

An interesting offshoot of the above work is the following theorem.

**Theorem 12.** *Let NAE2SATPOS denote the class of NA2SAT problems in which every literal is positive. NAE2SATPOS is* L-complete.

**Proof:** First observe that NAE2SATPOS is trivially in L, since it is a special case of NAE2SAT. Likewise, it has already been established that Undirected Graph 2-coloring (UG2COL) is in SL and therefore in L [3,8,4]. Now, consider the following $\mathtt{AC}_0$ reduction from UG2COL to NAE2SATPOS.

Let $G = \langle V, E\rangle$ denote an instance of the UG2COL problem, where, $V = \{v_1, v_2, \ldots, v_n\}$ and $E = \{e_{ij} :$ there is an undirected edge between vertices $v_i$ and $v_j\}$ .

We construct the following instance of NAESATPOS:

(a) Corresponding to vertex $v_i$, the variable $x_i$ is created.
(b) Corresponding to edge $e_{ij} = (v_i, v_j)$, the clause $(x_i, x_j)$ is created.
(c) The conjunction of all the clauses gives us the 2CNF formula $\phi(x)$.

It is not hard to see that the graph $G$ has a 2-coloring if and only if $\phi(x)$ is NAE-satisfiable. □

*This result can be seen as an addition to the collection of results in [9].*

## 6 Conclusion

In this paper, we designed and analyzed a randomized, literal-flipping algorithm for the NAE2SAT problem. As mentioned before, the existence of a polynomial time randomized algorithm for this problem is not surprising, since NAE2SAT belongs to the complexity class $\mathtt{SL} \subseteq \mathtt{P} \subseteq \mathtt{RP}$. The interesting aspect of our work is the simplicity of the randomized algorithm and its analysis. We extended the algorithm to check for bipartiteness in undirected graphs.

## References

1. Papadimitriou, C.H.: On selecting a satisfying truth assignment. In: IEEE (ed.) Proceedings: 32nd annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, October 1–4, pp. 163–169. IEEE Computer Society Press, USA (1991)
2. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman Company, San Francisco (1979)
3. Reif, J.H.: Symmetric complementation. J. ACM 31(2), 401–421 (1984)

4. Reingold, O.: Undirected st-connectivity in log-space. In: STOC, pp. 376–385 (2005)
5. Aspvall, B., Plass, M.F., Tarjan, R.: A linear-time algorithm for testing the truth of certain quantified boolean formulas. Information Processing Letters 8(3), 121–123 (1979)
6. Motwani, R., Raghavan, P.: Randomized Algorithms. Cambridge University Press, Cambridge (1995)
7. Ross, S.M.: Probability Models, 7th edn. Academic Press, Inc., London (2000)
8. Àlvarez, C., Greenlaw, R.: A compendium of problems complete for symmetric logarithmic space. Electronic Colloquium on Computational Complexity (ECCC) 3(39) (1996)
9. Johannsen, J.: Satisfiability problems complete for deterministic logarithmic space. In: STACS, pp. 317–325 (2004)