Electronic Colloquium on Computational Complexity, Report No. 10 (2003)

Improving a probabilistic 3-SAT Algorithm by Dynamic Search and Independent Clause Pairs

Sven Baumer* and Rainer Schuler

Abt. Theoretische Informatik, Universität Ulm, D-89069 Ulm, Germany

Abstract The satisfiability problem of Boolean Formulae in 3-CNF (3-SAT) is a well known NP-complete problem and the development of faster (moderately exponential time) algorithms has received much interest in recent years. We show that the 3-SAT problem can be solved by a probabilistic algorithm in expected time $O(1,3290^n)$. Our approach is based on Schöning's random walk algorithm for k-SAT, modified in two ways.

Keywords Probabilistic algorithm, Satisfiability, 3-SAT, Complexity.

1 Introduction

The satisfiability problem of Boolean formulae is one of the best known NP-complete problems. The problem remains NP-complete if the formulae are restricted to conjunctive normal form, where each clause contains exactly 3 literals (3-CNF). That is, polynomial time algorithms to test whether a formula in 3-CNF is satisfiable do not exists, unless P=NP.

In recent years, many algorithms have been developed which improve upon the trivial 2^n time complexity, where n is the number of variables in the formula. Milestones in the search for better SAT-algorithms are [MS85,PPZ97,Sch92,Kul99,PPSZ98], with respective run-times of $O(1,6181^n), O(1,588^n), O(1,579^n), O(1,505^n)$ and $O(1,447^n)$. [PPSZ98] also announces an improvement to $O(1,362^n)$. [Rod96] claims a run-time of $O(1,476^n)$.

In [Sch99] Schöning proposed a probabilistic algorithm to solve the 3-SAT problem. He showed that the success probability of the algorithm is $(\frac{3}{4})^n \cdot \frac{1}{p(n)}$ for some polynomial $p(n)$ in the worst-case, and accordingly has an expected run-time of $O(1,3334^n)$. The algorithm consists of two basic steps:

> An initial assignment is chosen randomly, where each variable is set to 0 or 1 independently with probability $\frac{1}{2}$.
> In each of following $3n$ steps the truth assignment of one of the variables is flipped (changed). The variable is chosen randomly with probability $\frac{1}{3}$ from a clause which is not satisfied by the current assignment.
> The algorithm succeeds if all clauses are satisfied.

Consider a (satisfiable) formula F and let a^* denote a fixed satisfying assignment. In the first step of the algorithm, an initial assignment a is chosen randomly. In each iteration of the algorithm, the Hamming distance $d(a,a^*)$ between the current assignment a and a^* decreases by 1 with probability at least $\frac{1}{3}$ and increases by 1 with probability at most $\frac{2}{3}$.

* supported by DFG-project Scho 302/5-2

ISSN 1433-8092

A careful analysis [Sch99] using Markov chains shows that the probability to reach a satisfying assignment from an initial assignment a is at least $(\frac{1}{2})^{d(a,a^*)} \cdot \frac{1}{p(n)}$ for some polynomial $p(n)$. Hence the success probability of the algorithm is bounded by $E[(\frac{1}{2})^{d(a,a^*)} \cdot \frac{1}{p(n)}] = E[(\frac{1}{2})^{d(a,a^*)}] \cdot \frac{1}{p(n)}$, where $E[Y]$ is the expected value of a random variable Y. In the following $p(n)$ will always denote the polynomial factor obtained as in [Sch99]. Further analysis then shows the overall success probability to find a satisfying assignment if one exists to be at least $Pr[\text{Success}] \geq (\frac{3}{4})^n \cdot \frac{1}{p(n)}$ in the case of formulae in 3-CNF, and more generally $(\frac{2k-2}{k})^n \cdot \frac{1}{p(n)}$ for k-CNF.

One way to improve this algorithm is to use a more sophisticated strategy for choosing the initial assignment. For example, it might be a good strategy to increase the probability of choosing initial assignments a with small $d(a, a^*)$, i.e. assignments already close to the satisfying assignment a^*. However, observe that the aim is not just to improve the average Hamming distance $E[d(a, a^*)]$, but instead the weighted average $E[(\frac{1}{2})^{d(a,a^*)}]$. To find such initial assignments, we exploit properties of the structure of the formula F, which is not done in the original algorithm [Sch99].

The first such approach to improve the success probability of Schöning's random walk algorithm was [HSSW02] by identifying 'independent clauses' in F, i.e. clauses that are variable disjoint. Hofmeister et al [HSSW02] show that the success probability of the algorithm can be improved if the initial assignment is chosen from assignments that satisfy the independent clauses. Note that the assignment of the variables can be chosen independently for each independent clause. Depending on the number of such clauses, one of two different algorithms is more likely to find a satisfying assignment: in the case of 'many' independent clauses the random walk algorithm will find a satisfying assignment, whereas in the case of 'few' independent clauses, a naive, brute-force search algorithm, which considers all assignments that satisfy the independent clauses, will succeed. Using this strategy, the expected run-time of the random walk algorithm is improved to $O(1,3302^n)$ [HSSW02].

We improve on the success probability in two ways. The approach of Hofmeister et al is extended in this paper by considering pairs of clauses which have (at least) one variable in common. Every clause contains three literals, i.e. restricts the number of assignments to the three variables that satisfy the clause to 7 (out of 8). If two clauses contain a common variable, the fraction of possible assignments decreases further, e.g. only 24 out of 32 assignments satisfy $\{x, y, z\}$ and $\{\neg x, u, v\}$.

A further improvement can be achieved, if a different strategy is used to identify the set of independent clause pairs. Instead of the greedy fashion of [HSSW02], we use a dynamic approach: depending on a partial assignment to the clause pairs already selected, new clause pairs are chosen. For example using this strategy only 6 (out of 8) assignments have to be considered for each independent clause. Again we can use a trade off between the success probability of the algorithm selecting the independent clause pairs (referred to as BF in the following, which increases if only few clause pairs are detected) and the success probability of the random walk algorithm (RW, which increases with the number of independent clause pairs). Using the dynamic strategy, the expected run-time to find a satisfying assignment is $O(1,3300^n)$ if only independent clauses are considered and improves to $O(1,3293^n)$ if the approach is extended to independent clause pairs. We also sketch a further improvement to $O(1,3290^n)$ by a more detailed case analysis.

2 An improvement of the brute force algorithm for independent clause selection

In the following let F denote a formula in 3-CNF, i.e. F is a conjunction of clauses where every clause contains (at most) 3 literals. Two clauses are called *independent* if the variables of the clauses are disjoint [ASTW01]. A set of clauses is called independent if the clauses are pairwise independent. A set of independent clauses is maximal, if every clause contains at least one variable which also occurs in some independent clause.

As shown in [HSSW02] the success probability of the random walk algorithm is $(\frac{3}{7})^m \cdot (\frac{3}{4})^{n-3m} \cdot \frac{1}{p(n)}$ and increases with the number m of independent clauses if the initial assignment is chosen randomly from assignments that satisfy the independent clauses. On the other hand, a brute force algorithm can consider every possible assignment to the variables of the independent clauses. If the set of independent clauses is maximal, the simplified formula will be in 2-CNF and its satisfiability can be checked in polynomial time [APT79]. Hence using exhaustive search satisfiability can be checked in time 7^m. In the following, a randomized variant of the exhaustive search algorithm is used: For each clause, one of the 7 possible assignments is chosen uniformly at random. This yields a success probability of at least $\frac{1}{7}^m$, and an expected run-time of $E[7^m]$.

That is, if the number of independent clauses is sufficiently small, the success probability of the brute force algorithm (we assume the assignments are chosen randomly) will exceed the success probability of the random walk. The combined success probability will be minimal if $(\frac{1}{7})^m = (\frac{3}{7})^m \cdot (\frac{3}{4})^{n-3m} \cdot \frac{1}{p(n)}$.

In this section we will show that the success probability can be improved further if the independent clauses are selected more carefully. We show that it is sufficient to consider only 6 assignments for each independent clause. This gives a success probability of a brute force algorithm of $(\frac{1}{6})^m$, where m is the number of independent clauses identified on any accepting path.

The algorithm is based on the following simple observation. Each variable x occurs positive as well as negative (otherwise the truth assignment can be fixed to satisfy all occurrences). Assume the truth value of x is chosen randomly. If we choose $x = 0$, only 3 assignments are possible to the remaining two literals of the clause where x occurs positive (otherwise the clause would not be satisfied). Analogously, if we choose $x = 1$, only 3 assignments are possible for the remaining two literals of the clause where x occurs negative. The success probability (i.e. the probability that the correct assignment is chosen) is $\frac{1}{2} \cdot \frac{1}{3} = \frac{1}{6}$ in both cases. Furthermore, in each case a clause can be selected as an independent clause, since all occurrences of the three variables involved are eliminated by fixing the truth values of the according literals.

While this observation is true if all clauses contain exactly three literals, we also need to consider implications due to clauses in 2-CNF (referred to as *chains* in the following). For example, the clauses $\{\mathbf{x}, y, z\}, \{\neg x, u\}, \{\neg \mathbf{u}, v, w\}$ are dependent on x in the same way as $\{\mathbf{x}, y, z\}, \{\neg \mathbf{x}, v, w\}$ are. The procedure is shown in more detail in Figure 1.

Throughout, we assume that the truth values of literals in unit clauses (i.e. clauses that only contain one literal) will be fixed accordingly. Furthermore, we assume that the truth value of a variable x will be set to $a \in \{0, 1\}$, if the partial assignment $x = \neg a$ leads to a contradiction (i.e. a clause will be empty/falsified).

Repeat
choose a variable x
find a clause $C = \{l_1, l_2, l_3\}$ such that $(x = 0)$ implies $(l_1 = 0)$, i.e.
$l_1 = x$ or there exist clauses $D_i = \{d_i, \neg d_{i+1}\}$, $1 \leq i \leq k$, and
$d_1 = x$ and $l_1 = d_{k+1}$
if C does not exist deterministically set $x = 0$, otherwise:
find a clause $C' = \{l'_1, l'_2, l'_3\}$ such that $(x = 1)$ implies $(l'_1 = 0)$, i.e.
$l'_1 = \neg x$ or there exist clauses $D'_i = \{\neg d'_i, d'_{i+1}\}$, $1 \leq i \leq k$, and
$d'_1 = \neg x$ and $l'_1 = \neg d_{k+1}$
if C' does not exist deterministically set $x = 1$, otherwise:
randomly guess truth assignment x from $\{0, 1\}$
if $x = 0$ then
randomly guess truth assignment of l_2, l_3 from $\{01, 10, 11\}$
select C as independent clause
if $x = 1$ then
randomly guess truth assignment of l'_2, l'_3 from $\{01, 10, 11\}$
select C' as independent clause

Figure 1. Improved brute force algorithm (BF) for independent clause selection.

Claim 1. Assume that a formula F is satisfiable. Then the algorithm BF will find a satisfying assignment with positive probability.

Observe that the following is true if x is set to 0 deterministically: $\neg x$ does not appear in any clause containing 3 literals. Furthermore, all clauses with 2 literals containing x can be satisfied by assigning 1 to the second literal. Moreover, the literals whose truth assignments are forced by a chain of such 2 literal-clauses do also not appear negatively in any clause with 3 literals. That is, all clauses containing x (positive or negative) or a forced variable are true under this assignment; either because the forced literal occurs positively, or the second literal is also forced to 1 (also note that if this process leads to a contradiction then x is forced to 1 by the simplification step). In particular, if F is satisfiable, then the formula with x set to 0 is also satisfiable.

Claim 2. Assume that a formula F is satisfiable and let a^* denote a satisfying assignment found on some path of the algorithm BF. The probability that a satisfying assignment is found is at least 6^{-m}, where m is the number of independent clauses detected on a path corresponding to the assignment a^*. Furthermore, the probability that $l \leq m$ independent clauses are detected is at least 6^{-l}.

The Claim follows from the observation that in each step of the procedure the truth assignment of C or C' will be chosen correctly (according to a satisfying assignment a^*) with probability $\frac{1}{6} = \frac{1}{2} \cdot \frac{1}{3}$.

The brute force algorithm will be combined with the random walk RW [HSSW02] in an obvious way.

Repeat
let $\mathcal{D} = \emptyset$
run BF and
let $\mathcal{C}$ denote the set of indep. clauses detected during the computation

IF $\|\mathcal{D}\| < \|\mathcal{C}\|$ then $\mathcal{D} = \mathcal{C}$
run RW using $\mathcal{D}$ as the set of independent clauses.

Theorem. The algorithm has expected running time of $O(1,3300^n)$.

The success probability of RW depends on the number of independent clauses on a path to a satisfying assignment. Let c denote some constant. We distinguish two cases.

1. The number of independent clauses on some path of the algorithm BF is less than $c \cdot n$.
   In this case the algorithm BF will find a satisfying assignment in expected time 6^{cn}.
2. The number of independent clauses is larger than $c \cdot n$.
   In this case the algorithm BF will find $\geq c$ independent clauses in expected time 6^{cn}. Afterwards the RW algorithm will find a satisfying assignment in expected time $(7/3)^c \cdot (4/3)^{n-3c} \cdot \frac{1}{p(n)}$.

Choose c such that $6^{cn} = (7/3)^{cn} \cdot (4/3)^{n-3cn} \cdot \frac{1}{p(n)}$, i.e.

$$c = \frac{\log(4/3)}{\log 6 - \log(7/3) + 3\log(4/3)} \approx 0.1591, \quad \text{and } 6^{cn} = 1.3300^n$$

A further improvement on the number of independent clauses might be possible. We note however, that in this way the success probability of the random walk algorithm cannot be improved beyond 0.7539^n which corresponds to an expected run-time of $(7/3)^{n/3} \approx 1,3264^n$.

In the next section we show how the bound in the theorem above can be further improved if the notion of independent clauses is extended to clause pairs.

3 Independent clause pairs

We will show that it is possible to choose the initial assignment for the RW algorithm from assignments that satisfy pairs of clauses which overlap with one or more variables. We show that the success probability of the random walk algorithm improves further with the number of such clause pairs detected.

Recall two clauses are independent if the variables of the clauses are disjoint [ASTW01]. The notion of independent clauses can be extended to clause pairs, or in general to clause tuples. Two clauses that share at least one variable are said to *overlap*. In the following we use *clause pair* to refer to a pair of clauses that overlap. Two clause pairs are called independent, if the sets of variables of the two clause pairs are disjoint. A set of clause pairs is called independent if the clause pairs are mutually independent.

A clause pair can overlap with 1, 2 or 3 literals, where some of the literals may be negated in one clause. Possible overlaps with a clause $\{x, y, z\}$ are shown in the following table, where u, v, w denote variables different from x, y, z. We use the following notation to describe the type of overlap: $-$ indicates a variable with negative overlap, $+$ indicates a positive overlap, and $*$ indicates a new variable not contained in the first clause. Observe that $(***)$ corresponds to two independent clauses. Also note that the case $(+++)$ is not possible since two clauses that agree on all three literals would be identical.

$(***)$	$(+**)$	$(-**)$	$(++*)$	$(--*)$	$(+-*)$	$(---)$	$(++-)$	$(+--)$
$\{u,v,w\}$	$\{x,u,v\}$	$\{\neg x,u,v\}$	$\{x,y,v\}$	$\{\neg x,\neg y,v\}$	$\{x,\neg y,v\}$	$\{\neg x,\neg y,\neg z\}$	$\{x,y,\neg z\}$	$\{x,\neg y,\neg z\}$

Let r_i denote the number of clauses and clause pairs of type i detected (e.g. by some greedy method). In particular we assume that the set of independent clauses and clause pairs is maximal, i.e. all other clauses share at least one variable with a selected clause or clause pair. Let $\hat{n} := n - \sum_i r_i$ denote the number of the remaining variables.

Now we consider the random walk algorithm, where the initial assignment is chosen randomly as follows. For each clause pair and each independent clause the truth assignment to the variables is chosen independently from the set of assignments that satisfy the clause pair and clause, respectively; the exact probabilities are given below. The truth values of the remaining $\hat{n}$ variables are chosen independently from $\{0,1\}$ with probability 1/2.

Let a^* denote a fixed satisfying assignment of F and a denote the randomly chosen initial assignment. Define $W_i^{(j)}$ to be the random variable which is the Hamming distance between a^* and a on the variables of the j-th independent clause or clause pair of type i. Similar, define V to be the random variable which is the Hamming distance of the assignments a and a^* on the variables which are not contained in an independent clause or independent clause pair. The success probability of the algorithm can be calculated using the inequality [Sch99,HSSW02]

$$\text{P[success]} \geq E[(\frac{1}{2})^{d(a,a^*)}] \cdot \frac{1}{p(n)}$$

The Hamming distance can be rewritten as: $d(a,a^*) = V + \sum_i \sum_{j=1}^{r_i} W_i^{(j)}$, where i ranges over all types of overlaps (see table above). The random variables $W_i^{(j)}$ are independent and equally distributed for each i. Since, as shown in section 3.3, $E[(\frac{1}{2})^{W_i^{(j)}}]$ does not depend on a^*, it follows that $E[(\frac{1}{2})^{W_i^{(j)}}] =: E[(\frac{1}{2})^{W_i}]$ is independent of j. Using $E[(\frac{1}{2})^V] = (\frac{3}{4})^{\hat{n}} \cdot \frac{1}{p(n)}$ [Sch99], we get

$$\text{P[success]} \geq E[(\frac{1}{2})^V] \cdot \Pi_i \Pi_{j=1}^{r_i} E[(\frac{1}{2})^{W_i^{(j)}}] \cdot \frac{1}{p(n)} \;=\; (\frac{3}{4})^{\hat{n}} \cdot \Pi_i (E[(\frac{1}{2})^{W_i}])^{r_i} \cdot \frac{1}{p(n)} \qquad (1)$$

We note that for every partial truth assignment to the variables of the independent clause pairs and independent clauses, the remaining formula will be in 2-CNF. The running time (or success probability) of a brute force algorithm will depend on the number of independent clauses and the number and type of the independent clause pairs. We consider a generalization (see Figure 2) of the brute force algorithm which selects a maximal set of independent clauses and clause pairs.

3.1 Clause pair selection

The idea is to extend the independent clause selection method (Figure 1) one step further. Let F be a satisfiable formula in 3-CNF. Each variable x occurs both positively and negatively in some clauses (containing 3 literals). Let a^* denote a satisfying assignment. The truth assignment of x is either 0 and 1. In both cases the truth assignment of the remaining two literals of one of the clauses is 01, 10, or 11. That is, at least one of the two literals will be set to 1. Since this literal also occurs positively and negatively, we will find some additional clause (containing 3 literals), where this literal occurs negatively. In this clause, the negated literal is set to 0, and the truth assignment of the remaining literals can be chosen randomly from assignments that satisfy the clause. In this way, the truth assignment of independent clause pairs can be guessed correctly with high probability.

For example, if $\{x, u, v\}$ and $\{\neg u, w, r\}$ are the second and third clause, the assignment to three variables x, u, v is correct with probability $\frac{1}{2} \cdot \frac{1}{3}$ and the assignment to the remaining two variables w, r is correct with probability $\frac{1}{3}$. In this case a correct assignment to five variables is chosen with probability $\frac{1}{2} \cdot \frac{1}{3} \cdot \frac{1}{3} = \frac{1}{18}$, while a pair of overlapping clauses is selected.

On the other hand, we will show that choosing an initial assignment that satisfies clause pairs will guarantee an improved success probability of the random walk algorithm. The exact probabilities for the assignments are calculated in Section 3.3 We will need to distinguish the different types of overlap, depending on the number of common variables, and depending on their *sign* (which is called positive if the variable or the negated variable occurs in both clauses, and called negative if the variable occurs in one clause and the negated variable occurs in the other clause).

The careful reader will notice that the idea could be further extended to clause triplets or clause n-tuples in general, but the calculations involved get rather complex while the benefit seems only minimal.

Repeat
  choose a variable x
  find clauses $C = \{l_1, l_2, l_3\}$ and $C' = \{l'_1, l'_2, l'_3\}$ such that
    $(x = 0)$ implies $(l_1 = 0)$ and $(x = 1)$ implies $(l'_1 = 0)$
  if C [respectively C'] does not exist set $x = 0$ [resp. $x = 1$], otherwise
  find $E = \{l_4, l_5, l_6\}$, $F = \{l_7, l_8, l_9\}$, $E' = \{l'_4, l'_5, l'_6\}$, $F' = \{l'_7, l'_8, l'_9\}$ such that
    $(l_2 = 1)$ implies $(l_4 = 0)$, and $(l_3 = 1)$ implies $(l_7 = 0)$, and
    $(l'_2 = 1)$ implies $(l'_4 = 0)$, and $(l'_3 = 1)$ implies $(l'_7 = 0)$
  if E [F, E' or F'] does not exist set $l_2 = 1$ [$l_3 = 1$, $l'_2 = 1$, or $l'_3 = 1$], otherwise
  randomly guess truth assignment of x from $\{0, 1\}$
  if $x = 0$ then
    randomly guess truth assignment of l_2, l_3 from $\{01, 10, 11\}$
    if $l_2 = 1$ then
      randomly guess truth assignment of the remaining literals of E
      select (C, E) as independent clause pair or independent clauses, resp.
    else if $l_3 = 1$ then
      randomly guess truth assignment of the remaining literals of F
      select (C, F) as independent clause pair or independent clauses, resp.
  if $x = 1$ then ... (analogously) ...

Figure 2. Brute force algorithm (BF) for independent clause pair selection.

The success probability of the brute force algorithm (Figure 2) depends on the number and type of the independent clauses and independent clause pairs detected. We have to distinguish between the cases that the clause pair (C, E) is selected via a *direct* overlap (i.e. $\neg l_2 \in E$) and via a *chain* of 2-CNF clauses. Observe that all types of direct overlap require at least one negatively shared variable (denoted –). Also observe that all chain cases contain at least one *, which denotes the literal which is forced to be 0 by the 2-CNF chain. All other literals denoted by a * are free and can be set to either 0 or 1.

The different types detected are given in the table below. In the second row we give the probability that the correct assignment is chosen by the BF algorithm, both for the direct and chain case. In the third row we give the success probability of the random walk algorithm RW, where the initial assignments are chosen optimally for each type of clause pairs (exact probabilities are given in Section 3.3 below).

In the last row the expected running time of the combined methods is given, if all variables are of the respective type. These running times are obtained by setting the success probabilities of BF and RW to be equal and using $E[\text{time}] = 1/Pr[\text{success}]$:

$$Pr[\text{BF success}]^m = Pr[\text{RW success}]^m \cdot (\frac{3}{4})^{n-m\cdot(\#\text{ variables})} \cdot \frac{1}{p(n)}$$

where m is the total number of clause pairs of the respective type.

type	$(***)$	$(+**)$	$(-**)$	$(++*)$	$(--*)$	$(+-*)$	$(---)$	$(++-)$	$(+--)$
# variables	3+3	5	5	4	4	4	3	3	3
BF direct	—	—	1/18	—	1/12	1/12	1/6	1/6	1/6
BF chain	1/18	1/12	1/12	1/6	1/6	1/6	—	—	—
RW	$(3/7)\cdot(3/7)$	81/331	27/110	27/83	15/46	15/46	7/16	9/20	45/104
time direct	—	—	$\mathbf{1, 3290^n}$	—	$1, 3288^n$	$1, 3288^n$	$1, 3258^n$	$1, 3201^n$	$1, 2941^n$
time chain	$\mathbf{1, 3293^n}$	$1, 3287^n$	$1, 3283^n$	$1, 3276^n$	$1, 3271^n$	$1, 3271^n$	—	—	—

The run-time of our algorithm is bounded by the maximum of these run-times, $1, 3293^n$. The worst case is when all clause pairs identified by the BF algorithm are of the type chain $(***)$, i.e. the two selected clauses are two independent clauses for the RW algorithm.

Of course, in practice the BF algorithm will not only identify clause pairs of one single type but a combination of different types. However, such 'mixed' cases have higher success probability and yield a better run-time than the worst case, where all clause pairs are of the same type (also see equation 1).

3.2 Sketch of a further improvement

Furthermore, we will show that the algorithm BF can be modified to improve in the worst case $(***)$ by repeating the previous step once more; this way instead of two independent clauses, either an independent clause and a pair or three independent clauses are selected. This way, we obtain a slightly better run-time of $1, 3290^n$ (also highlighted in the above table).

Assume that clauses C, E, F and C', E', F' are detected by the algorithm and that $C = \{l_1, l_2, l_3\}$ and $E = \{l_4, l_5, l_6\}$ are independent, i.e. do not share a variable. W.l.o.g. we assume that $x = 0$ and that $l_2 l_3 \in \{10, 11\}$ and hence (C, E) will be chosen as independent clauses. Before choosing the assignment of l_5, l_6 we test each of the three possible assignments $l_5 l_6 \in \{01, 10, 11\}$. If the assignment leads to a contradiction it is not considered any more. Otherwise we test whether the assignment is in fact admissible, that is we test whether it does not change the satisfiability of F. We look for a clause G with 3 literals such that a implies that one of the literals of G is set to 0. If G does not exist, then $l_4 l_5$ is set to a.

We modify the algorithm to distinguish the three cases. (i) If all assignments lead to a contradiction then the algorithm fails. (ii) If one of the assignments is admissible, we choose C and E as independent clauses. The probability that the correct assignment is chosen

for the literals of C and E is $\frac{1}{6}$. (iii) Otherwise we extend the algorithm. The assignment of $l_5 l_6$ is chosen randomly (from the assignments that do not lead to a contradiction). The remaining literals of the resp. clause G are chosen randomly from assignments that satisfy G. The clauses C, E and G are selected. C will be an independent clause, E and G will be an independent clause pair if E and G overlap, otherwise E and G will also be independent clauses.

In the case of three independent clause pairs the probability to choose the correct assignment to the 9 variables will be $\frac{1}{6} \cdot \frac{1}{3} \cdot \frac{1}{3} = \frac{1}{54}$. Otherwise the probability for the correct assignment will be $\frac{1}{3} \cdot p$, where p denotes the probability that an independent clause pair of the same type is chosen by the (original) algorithm directly.

The success probability is minimal if 3 independent clauses are detected. Using $(1/54)^m = (3/7)^{3m} \cdot (3/4)^{n-9m}$ we obtain a run-time of $1,3289^n$ for this case. Hence the case $(- * *)$ is now the worst case, with a corresponding run-time of $O(1,3290^n)$.

3.3 Calculation of RW success probabilities

First we consider the case that the two clauses (C, E) share exactly one variable.

1. **Case (+ * *)**, i.e. the two clauses are of type $\{x, y, z\}, \{x, u, v\}$.
   The probability for the initial assignment should maximize the expected value $E[(1/2)^X]$ where X is the Hamming distance between the initial assignment a and a satisfying assignment a^*. Since the satisfying assignment a^* is unknown, we choose the probabilities that the expected value $E[(1/2)^X]$ is identical for all possible cases of a^*.
   To simplify the corresponding equation system, the 32 possible assignments to the literals x, y, z, u, v can be grouped into 12 equivalence classes. All assignments of a class have the same Hamming distances to assignments of the same and any other class. [1]
   In the following we will denote assignments to these five variables in the format $yzxuv$, i.e. the middle bit represents the truth value of the common variable. E.g. 00100 represents the truth assignment that sets x to 1, and the other variables to 0.
   Intuitively, all "mirror images" of an assignment are in the same class, e.g. assignments 00001, 00010, 01000 and 1000 all belong to the same class K_{11}. Note that literal x has a different status, therefore the assignment 00100 belongs to a different group (K_1). The 12 groups are as follows:

K_1	K_2	K_3	K_4	K_5	K_6	K_7	K_8	K_9	K_{10}	K_{11}	K_{12}
00100	00101	00111	01001	01011	01101	01111	11011	11111	00000	00001	00011
	00110	11100	01010	10011	01110	10111				00010	11000
	01100		10001	11001	10101	11101				01000	
	10100		10010	11010	10110	11110				10000	

   Observe that K_1 to K_9 consist of assignments that satisfy both clauses, whereas assignments from K_{10} to K_{12} don't.
   The (fixed) satisfying assignment a^* is from K_1 to K_9. Analyzing each case in the same fashion as [HSSW02] yields one equation for every choice of a^*. The fact that the probabilities add up to one yields the 10th equation.

[1] Observe that this classification is not a necessary step. We only choose to do so in order to simplify the equation system for presentation. Instead a system with 32 equations could be solved, yielding the same result.

Let p_i denote the probabilities that the 5 literals are set according to group K_i. For every choice of a^* the coefficients α_i of p_i, $1 \leq i \leq 9$ are given in the table below. Assume a^* is fixed. The coefficients are calculated as follows:

$$E[(\frac{1}{2})^{W_{(+**)}}] = \sum_{k=0}^{5} (\frac{1}{2})^k \sum_{i=1}^{9} p_i \cdot \|\{a \in K_i \mid d(a,a^*) = k\}\|$$
$$= \sum_{i=1}^{9} p_i \cdot \sum_{k=0}^{5} (\frac{1}{2})^k \|\{a \in K_i \mid d(a,a^*) = k\}\| = \sum_{i=1}^{9} \alpha_i \cdot p_i$$

a^*	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}	$=$
K_1	$\frac{32}{32}$	$\frac{64}{32}$	$\frac{16}{32}$	$\frac{16}{32}$	$\frac{8}{32}$	$\frac{32}{32}$	$\frac{16}{32}$	$\frac{1}{32}$	$\frac{2}{32}$	$\frac{16}{32}$	$\frac{32}{32}$	$\frac{8}{32}$	$E[(\frac{1}{2})^{W_{(+**)}}]$
K_2	$\frac{16}{32}$	$\frac{56}{32}$	$\frac{20}{32}$	$\frac{20}{32}$	$\frac{13}{32}$	$\frac{40}{32}$	$\frac{26}{32}$	$\frac{2}{32}$	$\frac{4}{32}$	$\frac{8}{32}$	$\frac{28}{32}$	$\frac{10}{32}$	$E[(\frac{1}{2})^{W_{(+**)}}]$
K_3	$\frac{8}{32}$	$\frac{40}{32}$	$\frac{34}{32}$	$\frac{16}{32}$	$\frac{20}{32}$	$\frac{32}{32}$	$\frac{40}{32}$	$\frac{4}{32}$	$\frac{8}{32}$	$\frac{4}{32}$	$\frac{20}{32}$	$\frac{17}{32}$	$E[(\frac{1}{2})^{W_{(+**)}}]$
K_4	$\frac{4}{32}$	$\frac{20}{32}$	$\frac{8}{32}$	$\frac{50}{32}$	$\frac{40}{32}$	$\frac{25}{32}$	$\frac{20}{32}$	$\frac{8}{32}$	$\frac{4}{32}$	$\frac{8}{32}$	$\frac{40}{32}$	$\frac{16}{32}$	$E[(\frac{1}{2})^{W_{(+**)}}]$
K_5	$\frac{2}{32}$	$\frac{13}{32}$	$\frac{10}{32}$	$\frac{40}{32}$	$\frac{56}{32}$	$\frac{20}{32}$	$\frac{28}{32}$	$\frac{16}{32}$	$\frac{8}{32}$	$\frac{4}{32}$	$\frac{26}{32}$	$\frac{20}{32}$	$E[(\frac{1}{2})^{W_{(+**)}}]$
K_6	$\frac{8}{32}$	$\frac{40}{32}$	$\frac{16}{32}$	$\frac{25}{32}$	$\frac{20}{32}$	$\frac{50}{32}$	$\frac{40}{32}$	$\frac{4}{32}$	$\frac{8}{32}$	$\frac{4}{32}$	$\frac{20}{32}$	$\frac{8}{32}$	$E[(\frac{1}{2})^{W_{(+**)}}]$
K_7	$\frac{4}{32}$	$\frac{26}{32}$	$\frac{20}{32}$	$\frac{20}{32}$	$\frac{28}{32}$	$\frac{40}{32}$	$\frac{56}{32}$	$\frac{8}{32}$	$\frac{16}{32}$	$\frac{2}{32}$	$\frac{13}{32}$	$\frac{10}{32}$	$E[(\frac{1}{2})^{W_{(+**)}}]$
K_8	$\frac{1}{32}$	$\frac{8}{32}$	$\frac{8}{32}$	$\frac{32}{32}$	$\frac{64}{32}$	$\frac{16}{32}$	$\frac{32}{32}$	$\frac{32}{32}$	$\frac{16}{32}$	$\frac{2}{32}$	$\frac{16}{32}$	$\frac{16}{32}$	$E[(\frac{1}{2})^{W_{(+**)}}]$
K_9	$\frac{2}{32}$	$\frac{16}{32}$	$\frac{16}{32}$	$\frac{16}{32}$	$\frac{32}{32}$	$\frac{32}{32}$	$\frac{64}{32}$	$\frac{16}{32}$	$\frac{32}{32}$	$\frac{1}{32}$	$\frac{8}{32}$	$\frac{8}{32}$	$E[(\frac{1}{2})^{W_{(+**)}}]$
$\sum$	1	4	2	4	4	4	4	1	1	1	4	2	1

Note that this is an under-determined equation system, with 10 equations (K_1 to K_9 and $\sum$) and 13 variables (p_1 to p_{12} and $E[(\frac{1}{2})^{W_{(+**)}}]$).
Setting $p_{10} = p_{11} = p_{12} = 0$ and solving the corresponding determined system using a computer algebra program (e.g. Maple 7) yields

$$E[(\frac{1}{2})^{W_{(+**)}}] = 81/331,$$

where the probabilities are given by

p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9
16/331	16/331	16/331	24/331	12/331	4/331	10/331	6/331	13/331

Solving the full, under-determined system proves the optimal value for p_{10}, p_{11}, p_{12} to be all 0, since positive values for these have a negative contribution to $E[(\frac{1}{2})^{W_{(+**)}}]$.

2. **Case (– * *)**, i.e. the two clauses are of type $\{x, y, z\}, \{\neg x, u, v\}$.
Observe that "mirror images" are also obtained by swapping the two left and right bits (y, z with u, v) and inverting x. Groups are as follows:

K_1	K_2	K_3	K_4	K_5	K_6	K_7	K_8	K_9
01000	01001	01011	00111	11001	11011	00000	00001	00011
10000	01010	10011	11000	11010	11111	00100	00010	11100
00101	10001	11101		01111			01100	
00110	10010	11110		10111			10100	
	01101							
	01110							
	10101							
	10110							

Solving the equation system

a^*	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	$=$
K_1	$\frac{48}{32}$	$\frac{60}{32}$	$\frac{18}{32}$	$\frac{18}{32}$	$\frac{21}{32}$	$\frac{6}{32}$	$\frac{24}{32}$	$\frac{36}{32}$	$\frac{12}{32}$	$E[(\frac{1}{2})^{W_{(-**)}}]$
K_2	$\frac{30}{32}$	$\frac{75}{32}$	$\frac{30}{32}$	$\frac{12}{32}$	$\frac{30}{32}$	$\frac{12}{32}$	$\frac{12}{32}$	$\frac{30}{32}$	$\frac{12}{32}$	$E[(\frac{1}{2})^{W_{(-**)}}]$
K_3	$\frac{18}{32}$	$\frac{60}{32}$	$\frac{48}{32}$	$\frac{12}{32}$	$\frac{36}{32}$	$\frac{24}{32}$	$\frac{6}{32}$	$\frac{21}{32}$	$\frac{18}{32}$	$E[(\frac{1}{2})^{W_{(-**)}}]$
K_4	$\frac{36}{32}$	$\frac{48}{32}$	$\frac{24}{32}$	$\frac{33}{32}$	$\frac{36}{32}$	$\frac{12}{32}$	$\frac{12}{32}$	$\frac{24}{32}$	$\frac{18}{32}$	$E[(\frac{1}{2})^{W_{(-**)}}]$
K_5	$\frac{21}{32}$	$\frac{60}{32}$	$\frac{36}{32}$	$\frac{18}{32}$	$\frac{48}{32}$	$\frac{24}{32}$	$\frac{6}{32}$	$\frac{18}{32}$	$\frac{12}{32}$	$E[(\frac{1}{2})^{W_{(-**)}}]$
K_6	$\frac{12}{32}$	$\frac{48}{32}$	$\frac{48}{32}$	$\frac{12}{32}$	$\frac{48}{32}$	$\frac{48}{32}$	$\frac{3}{32}$	$\frac{12}{32}$	$\frac{12}{32}$	$E[(\frac{1}{2})^{W_{(-**)}}]$
$\sum$	4	8	4	2	4	2	2	4	2	1

yields

$$E[(\frac{1}{2})^{W_{(-**)}}] = 27/110 > 81/331,$$

where the probabilities are given by

p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9
4/55	2/55	3/55	2/55	1/55	3/110	0	0	0

3. **Case (+ + *)**

K_1	K_2	K_3	K_4	K_5	K_6	K_7	K_8	K_9
0100	1001	0011	0110	1110	1101	1111	0000	0001
0010		0101		0111	1011			1000
		1100						
		1010						

a^*	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	$=$
K_1	$\frac{20}{16}$	$\frac{2}{16}$	$\frac{20}{16}$	$\frac{8}{16}$	$\frac{8}{16}$	$\frac{5}{16}$	$\frac{2}{16}$	$\frac{8}{16}$	$\frac{8}{16}$	$E[(\frac{1}{2})^{W_{(++*)}}]$
K_2	$\frac{4}{16}$	$\frac{16}{16}$	$\frac{16}{16}$	$\frac{1}{16}$	$\frac{4}{16}$	$\frac{16}{16}$	$\frac{4}{16}$	$\frac{4}{16}$	$\frac{16}{16}$	$E[(\frac{1}{2})^{W_{(++*)}}]$
K_3	$\frac{10}{16}$	$\frac{4}{16}$	$\frac{25}{16}$	$\frac{4}{16}$	$\frac{10}{16}$	$\frac{10}{16}$	$\frac{4}{16}$	$\frac{4}{16}$	$\frac{10}{16}$	$E[(\frac{1}{2})^{W_{(++*)}}]$
K_4	$\frac{16}{16}$	$\frac{1}{16}$	$\frac{16}{16}$	$\frac{16}{16}$	$\frac{16}{16}$	$\frac{4}{16}$	$\frac{4}{16}$	$\frac{4}{16}$	$\frac{4}{16}$	$E[(\frac{1}{2})^{W_{(++*)}}]$
K_5	$\frac{8}{16}$	$\frac{2}{16}$	$\frac{20}{16}$	$\frac{8}{16}$	$\frac{20}{16}$	$\frac{8}{16}$	$\frac{8}{16}$	$\frac{2}{16}$	$\frac{5}{16}$	$E[(\frac{1}{2})^{W_{(++*)}}]$
K_6	$\frac{5}{16}$	$\frac{8}{16}$	$\frac{20}{16}$	$\frac{2}{16}$	$\frac{8}{16}$	$\frac{20}{16}$	$\frac{8}{16}$	$\frac{2}{16}$	$\frac{8}{16}$	$E[(\frac{1}{2})^{W_{(++*)}}]$
K_7	$\frac{4}{16}$	$\frac{4}{16}$	$\frac{16}{16}$	$\frac{4}{16}$	$\frac{16}{16}$	$\frac{16}{16}$	$\frac{16}{16}$	$\frac{1}{16}$	$\frac{4}{16}$	$E[(\frac{1}{2})^{W_{(++*)}}]$
$\sum$	2	1	4	1	2	2	1	2	2	1

$$E[(\frac{1}{2})^{W_{(++*)}}] = 27/83$$

p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9
8/83	12/83	8/83	4/83	4/83	2/83	7/83	0	0

4. **Case (+ − *)**
   Observe that there is no grouping possible. The sixteen possible assignments fall into 16 groups. For simplicity, $K_0 = 0000$, $K_1 = 0001$, $K_2 = 0010$, ... , $K_{15} = 1111$. Also note that K_0, K_1, K_2 and K_{12} correspond to the non-satisfying assignments (0000, 0001, 0010, 1010).

a^*	p_0	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}	p_{13}	p_{14}	p_{15}	$=$
K_3	$\frac{4}{16}$	$\frac{8}{16}$	$\frac{8}{16}$	$\frac{16}{16}$	$\frac{2}{16}$	$\frac{4}{16}$	$\frac{4}{16}$	$\frac{8}{16}$	$\frac{2}{16}$	$\frac{4}{16}$	$\frac{4}{16}$	$\frac{8}{16}$	$\frac{1}{16}$	$\frac{2}{16}$	$\frac{2}{16}$	$\frac{4}{16}$	$E[(\frac{1}{2})^{W_{(+-*)}}]$
K_4	$\frac{8}{16}$	$\frac{4}{16}$	$\frac{4}{16}$	$\frac{2}{16}$	$\frac{16}{16}$	$\frac{8}{16}$	$\frac{8}{16}$	$\frac{4}{16}$	$\frac{4}{16}$	$\frac{2}{16}$	$\frac{2}{16}$	$\frac{1}{16}$	$\frac{8}{16}$	$\frac{4}{16}$	$\frac{4}{16}$	$\frac{2}{16}$	$E[(\frac{1}{2})^{W_{(+-*)}}]$
K_5	$\frac{4}{16}$	$\frac{8}{16}$	$\frac{2}{16}$	$\frac{4}{16}$	$\frac{8}{16}$	$\frac{16}{16}$	$\frac{4}{16}$	$\frac{8}{16}$	$\frac{2}{16}$	$\frac{4}{16}$	$\frac{1}{16}$	$\frac{2}{16}$	$\frac{4}{16}$	$\frac{8}{16}$	$\frac{2}{16}$	$\frac{4}{16}$	$E[(\frac{1}{2})^{W_{(+-*)}}]$
K_6	$\frac{4}{16}$	$\frac{2}{16}$	$\frac{8}{16}$	$\frac{4}{16}$	$\frac{8}{16}$	$\frac{4}{16}$	$\frac{16}{16}$	$\frac{8}{16}$	$\frac{2}{16}$	$\frac{1}{16}$	$\frac{4}{16}$	$\frac{2}{16}$	$\frac{4}{16}$	$\frac{2}{16}$	$\frac{8}{16}$	$\frac{4}{16}$	$E[(\frac{1}{2})^{W_{(+-*)}}]$
K_7	$\frac{2}{16}$	$\frac{4}{16}$	$\frac{4}{16}$	$\frac{8}{16}$	$\frac{4}{16}$	$\frac{8}{16}$	$\frac{8}{16}$	$\frac{16}{16}$	$\frac{1}{16}$	$\frac{2}{16}$	$\frac{2}{16}$	$\frac{4}{16}$	$\frac{2}{16}$	$\frac{4}{16}$	$\frac{4}{16}$	$\frac{8}{16}$	$E[(\frac{1}{2})^{W_{(+-*)}}]$
K_8	$\frac{8}{16}$	$\frac{4}{16}$	$\frac{4}{16}$	$\frac{2}{16}$	$\frac{4}{16}$	$\frac{2}{16}$	$\frac{2}{16}$	$\frac{1}{16}$	$\frac{16}{16}$	$\frac{8}{16}$	$\frac{8}{16}$	$\frac{4}{16}$	$\frac{8}{16}$	$\frac{4}{16}$	$\frac{4}{16}$	$\frac{2}{16}$	$E[(\frac{1}{2})^{W_{(+-*)}}]$
K_9	$\frac{4}{16}$	$\frac{8}{16}$	$\frac{2}{16}$	$\frac{4}{16}$	$\frac{2}{16}$	$\frac{4}{16}$	$\frac{1}{16}$	$\frac{2}{16}$	$\frac{8}{16}$	$\frac{16}{16}$	$\frac{4}{16}$	$\frac{8}{16}$	$\frac{4}{16}$	$\frac{8}{16}$	$\frac{2}{16}$	$\frac{4}{16}$	$E[(\frac{1}{2})^{W_{(+-*)}}]$
K_{10}	$\frac{4}{16}$	$\frac{2}{16}$	$\frac{8}{16}$	$\frac{4}{16}$	$\frac{2}{16}$	$\frac{1}{16}$	$\frac{4}{16}$	$\frac{2}{16}$	$\frac{8}{16}$	$\frac{4}{16}$	$\frac{16}{16}$	$\frac{8}{16}$	$\frac{4}{16}$	$\frac{2}{16}$	$\frac{8}{16}$	$\frac{4}{16}$	$E[(\frac{1}{2})^{W_{(+-*)}}]$
K_{11}	$\frac{2}{16}$	$\frac{4}{16}$	$\frac{4}{16}$	$\frac{8}{16}$	$\frac{1}{16}$	$\frac{2}{16}$	$\frac{2}{16}$	$\frac{4}{16}$	$\frac{4}{16}$	$\frac{8}{16}$	$\frac{8}{16}$	$\frac{16}{16}$	$\frac{2}{16}$	$\frac{4}{16}$	$\frac{4}{16}$	$\frac{8}{16}$	$E[(\frac{1}{2})^{W_{(+-*)}}]$
K_{13}	$\frac{2}{16}$	$\frac{4}{16}$	$\frac{1}{16}$	$\frac{2}{16}$	$\frac{4}{16}$	$\frac{8}{16}$	$\frac{2}{16}$	$\frac{4}{16}$	$\frac{4}{16}$	$\frac{8}{16}$	$\frac{2}{16}$	$\frac{4}{16}$	$\frac{8}{16}$	$\frac{16}{16}$	$\frac{4}{16}$	$\frac{8}{16}$	$E[(\frac{1}{2})^{W_{(+-*)}}]$
K_{14}	$\frac{2}{16}$	$\frac{1}{16}$	$\frac{4}{16}$	$\frac{2}{16}$	$\frac{4}{16}$	$\frac{2}{16}$	$\frac{8}{16}$	$\frac{4}{16}$	$\frac{4}{16}$	$\frac{2}{16}$	$\frac{8}{16}$	$\frac{4}{16}$	$\frac{8}{16}$	$\frac{4}{16}$	$\frac{16}{16}$	$\frac{8}{16}$	$E[(\frac{1}{2})^{W_{(+-*)}}]$
K_{15}	$\frac{1}{16}$	$\frac{2}{16}$	$\frac{2}{16}$	$\frac{4}{16}$	$\frac{2}{16}$	$\frac{4}{16}$	$\frac{4}{16}$	$\frac{8}{16}$	$\frac{2}{16}$	$\frac{4}{16}$	$\frac{4}{16}$	$\frac{8}{16}$	$\frac{4}{16}$	$\frac{8}{16}$	$\frac{8}{16}$	$\frac{16}{16}$	$E[(\frac{1}{2})^{W_{(+-*)}}]$
$\sum$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

$$E[(\frac{1}{2})^{W_{(+-*)}}] = 15/46$$

p_0	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}	p_{13}	p_{14}	p_{15}
0	0	0	10/69	8/69	2/23	2/23	2/69	8/69	2/23	2/23	2/69	0	5/69	5/69	5/69

5. **Case (− − *)**
   Grouping is possible, similar to the (+ + *) case:

K_1	K_2	K_3	K_4	K_5	K_6	K_7	K_8	K_9	K_{10}	K_{11}
0100	1000	0011	1001	0111	1101	1111	0000	0001	0110	1110
0010		0101			1011					
		1100								
		1010								

a^*	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	$=$
K_1	$\frac{20}{16}$	$\frac{4}{16}$	$\frac{20}{16}$	$\frac{2}{16}$	$\frac{4}{16}$	$\frac{5}{16}$	$\frac{2}{16}$	$\frac{8}{16}$	$\frac{4}{16}$	$\frac{8}{16}$	$\frac{4}{16}$	$E[(\frac{1}{2})^{W_{(--*)}}]$
K_2	$\frac{8}{16}$	$\frac{16}{16}$	$\frac{20}{16}$	$\frac{8}{16}$	$\frac{1}{16}$	$\frac{8}{16}$	$\frac{2}{16}$	$\frac{8}{16}$	$\frac{4}{16}$	$\frac{2}{16}$	$\frac{4}{16}$	$E[(\frac{1}{2})^{W_{(--*)}}]$
K_3	$\frac{10}{16}$	$\frac{2}{16}$	$\frac{25}{16}$	$\frac{4}{16}$	$\frac{8}{16}$	$\frac{10}{16}$	$\frac{4}{16}$	$\frac{4}{16}$	$\frac{8}{16}$	$\frac{4}{16}$	$\frac{2}{16}$	$E[(\frac{1}{2})^{W_{(--*)}}]$
K_4	$\frac{4}{16}$	$\frac{8}{16}$	$\frac{16}{16}$	$\frac{16}{16}$	$\frac{2}{16}$	$\frac{16}{16}$	$\frac{4}{16}$	$\frac{4}{16}$	$\frac{8}{16}$	$\frac{1}{16}$	$\frac{2}{16}$	$E[(\frac{1}{2})^{W_{(--*)}}]$
K_5	$\frac{8}{16}$	$\frac{1}{16}$	$\frac{20}{16}$	$\frac{2}{16}$	$\frac{16}{16}$	$\frac{8}{16}$	$\frac{8}{16}$	$\frac{2}{16}$	$\frac{4}{16}$	$\frac{8}{16}$	$\frac{4}{16}$	$E[(\frac{1}{2})^{W_{(--*)}}]$
K_6	$\frac{5}{16}$	$\frac{4}{16}$	$\frac{20}{16}$	$\frac{8}{16}$	$\frac{4}{16}$	$\frac{20}{16}$	$\frac{8}{16}$	$\frac{2}{16}$	$\frac{4}{16}$	$\frac{2}{16}$	$\frac{4}{16}$	$E[(\frac{1}{2})^{W_{(--*)}}]$
K_7	$\frac{4}{16}$	$\frac{2}{16}$	$\frac{16}{16}$	$\frac{4}{16}$	$\frac{8}{16}$	$\frac{16}{16}$	$\frac{16}{16}$	$\frac{1}{16}$	$\frac{2}{16}$	$\frac{4}{16}$	$\frac{8}{16}$	$E[(\frac{1}{2})^{W_{(--*)}}]$
$\sum$	2	1	4	1	1	2	1	1	1	1	1	1

$$E[(\frac{1}{2})^{W_{(--*)}}] = 15/46$$

p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}
8/69	5/69	2/23	5/46	5/69	2/69	5/46	0	0	0	0

Note that this the only case where assigning a positive probability to a non-satisfying assignment ($K_9 = 0001$) would further improve the expectancy $E[(\frac{1}{2})^{W_{(--*)}}] = 15/46 + \frac{3}{368} \cdot p_9$.

6. **Cases $(- - -)$, $(+ - -)$, $(+ + -)$ and $(* * *)$**
For the sake of brevity, these cases are solved simultaneously. Observe that $(* * *)$ corresponds to independent clauses, while the other cases correspond to clause pairs. The eight possible assignments fall into 8 groups. For simplicity, $K_0 = 000$, $K_1 = 001$, $\ldots$, $K_7 = 111$. Also observe that K_0 always corresponds to a non-satisfying assignment, and depending on the sub-case, so does some other group.

a^*	p_0	p_1	p_2	p_3	p_4	p_5	p_6	p_7	$=$
K_0	$\frac{8}{8}$	$\frac{4}{8}$	$\frac{4}{8}$	$\frac{2}{8}$	$\frac{4}{8}$	$\frac{2}{8}$	$\frac{2}{8}$	$\frac{1}{8}$	$E[(\frac{1}{2})^{W_{(???)}}]$
K_1	$\frac{4}{8}$	$\frac{8}{8}$	$\frac{2}{8}$	$\frac{4}{8}$	$\frac{2}{8}$	$\frac{4}{8}$	$\frac{1}{8}$	$\frac{2}{8}$	$E[(\frac{1}{2})^{W_{(???)}}]$
K_2	$\frac{4}{8}$	$\frac{2}{8}$	$\frac{8}{8}$	$\frac{4}{8}$	$\frac{2}{8}$	$\frac{1}{8}$	$\frac{4}{8}$	$\frac{2}{8}$	$E[(\frac{1}{2})^{W_{(???)}}]$
K_3	$\frac{2}{8}$	$\frac{4}{8}$	$\frac{4}{8}$	$\frac{8}{8}$	$\frac{1}{8}$	$\frac{2}{8}$	$\frac{2}{8}$	$\frac{4}{8}$	$E[(\frac{1}{2})^{W_{(???)}}]$
K_4	$\frac{4}{8}$	$\frac{2}{8}$	$\frac{2}{8}$	$\frac{1}{8}$	$\frac{8}{8}$	$\frac{4}{8}$	$\frac{4}{8}$	$\frac{2}{8}$	$E[(\frac{1}{2})^{W_{(???)}}]$
K_5	$\frac{2}{8}$	$\frac{4}{8}$	$\frac{1}{8}$	$\frac{2}{8}$	$\frac{4}{8}$	$\frac{8}{8}$	$\frac{2}{8}$	$\frac{4}{8}$	$E[(\frac{1}{2})^{W_{(???)}}]$
K_6	$\frac{2}{8}$	$\frac{1}{8}$	$\frac{4}{8}$	$\frac{2}{8}$	$\frac{4}{8}$	$\frac{2}{8}$	$\frac{8}{8}$	$\frac{4}{8}$	$E[(\frac{1}{2})^{W_{(???)}}]$
K_7	$\frac{1}{8}$	$\frac{2}{8}$	$\frac{2}{8}$	$\frac{4}{8}$	$\frac{2}{8}$	$\frac{4}{8}$	$\frac{4}{8}$	$\frac{8}{8}$	$E[(\frac{1}{2})^{W_{(???)}}]$
$\sum$	1	1	1	1	1	1	1	1	1

- Case $(- - -)$
  Solving the above equation system without equations K_0 and K_7 (000, 111) yields

$$E[(\frac{1}{2})^{W_{(---)}}] = 7/16$$

- Case $(+ - -)$
  Solving without equations K_0 and K_6 (000,110) yields

$$E[(\frac{1}{2})^{W_{(+--)}}] = 45/104$$

- Case $(+ + -)$
  Solving without equations K_0 and K_4 (000,100) yields

$$E[(\frac{1}{2})^{W_{(++-)}}] = 9/20$$

– Case $(***)$, i.e. an independent clause
Solving the above system without K_0 yields

$$E[(\frac{1}{2})^{W_{(***)}}] = 3/7$$

This corresponds to the strategy used in [HSSW02], which is improved upon in this paper.

Probabilities for the four cases are given below.

Case	p_0	p_1	p_2	p_3	p_4	p_5	p_6	p_7
$(---)$	0	1/6	1/6	1/6	1/6	1/6	1/6	0
$(+--)$	0	5/26	3/13	5/26	3/13	1/13	0	5/26
$(++-)$	0	1/5	1/5	1/10	0	1/5	1/5	1/10
$(***)$	0	4/21	4/21	2/21	4/21	2/21	2/21	1/7

References

[APT79] B. Aspvall, M.F. Plass, and R.E. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *IPL*, 8(3):121–123, 1979.

[ASTW01] S. Aida, R. Schuler, Tatsukiji, and O. Watanabe. On the difference between polynomial-time many-one and truth-table reducibilities on distributional problems. In *Proceedings of the 18th Annual Symposium on Theoretical Aspects of Computer Science*, 2001.

[HSSW02] T. Hofmeister, U. Schöning, R. Schuler, and O. Watanabe. A probabilistic 3-SAT algorithm further improved. In *Proceedings 19th Symposium on Theoretical Aspects of Computer Science*, volume 2285 of *LNCS*, pages 192–203. Springer-Verlag, 2002.

[Kul99] O. Kullmann. New methods for 3-sat decision and worst-case analysis. *Theoretical Computer Science*, 223:1–72, 1999.

[MS85] B. Monien and E. Speckenmeyer. Solving satisfiability in less than 2^n steps. *Discrete Applied Mathematics*, 10:287–295, 1985.

[PPZ97] R. Paturi, P. Pudlák and F. Zane. Satisfiability coding lemma. In *38th Ann. IEEE Sympos. on Foundations of Comp. Sci. (FOCS'97)*, pages 566–574, 1997.

[PPSZ98] R. Paturi, P. Pudlák, M.E. Saks, and F. Zane. An improved exponential-time algorithm for k-sat. In *39th Ann. IEEE Sympos. on Foundations of Comp. Sci. (FOCS'98)*, pages 410–414, 1998.

[Rod96] R. Rodosek. A new approach on solving 3-satisfiability. In *Proceedings 3rd Intern. Conf. on AI and Symbolic Math. Computation*, volume 1138 of *LNCS*, pages 197–212. Springer, 1996.

[Sch92] I. Schiermeyer. Solving 3-satisfiability in less than $o(1,579^n)$ steps. In *Computer Science Logic (CSL)*, volume 702 of *LNCS*, pages 379–394. Springer, 1992.

[Sch99] Uwe Schöning. A probabilistic algorithm for k-SAT and constraint satisfaction problems. In *Proc. 40th FOCS*, pages 410–414. ACM, 1999.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'