

Optimal myopic algorithms for random 3-SAT

Dimitris Achlioptas
Microsoft Research
One Microsoft Way
Redmond WA 98052

Gregory B. Sorkin
Department of Mathematical Sciences
IBM T.J. Watson Research Center
Yorktown Heights NY 10598

Abstract

Let $F_3(n, m)$ be a random 3-SAT formula formed by selecting uniformly, independently, and with replacement, m clauses among all $8\binom{n}{3}$ possible 3-clauses over n variables. It has been conjectured that there exists a constant r_3 such that for any $\epsilon > 0$, $F_3(n, (r_3 - \epsilon)n)$ is almost surely satisfiable, but $F_3(n, (r_3 + \epsilon)n)$ is almost surely unsatisfiable. The best lower bounds for the potential value of r_3 have come from analyzing rather simple extensions of unit-clause propagation. Recently, it was shown [2] that all these extensions can be cast in a common framework and analyzed in a uniform manner by employing differential equations. Here, we determine optimal algorithms expressible in that framework, establishing $r_3 > 3.26$. We extend the analysis via differential equations, and make extensive use of a new optimization problem we call “max-density multiple-choice knapsack”. The structure of optimal knapsack solutions elegantly characterizes the choices made by an optimal algorithm.

1 Introduction

Let $F_k(n, m)$ be a random k -SAT formula formed by selecting uniformly, independently, and with replacement, m clauses among all $2^k\binom{n}{k}$ non-trivial clauses of length k , i.e., clauses with k distinct, non-complementary literals¹. Let us say that a sequence of events \mathcal{E}_n holds with high probability (w.h.p.) if $\lim_{n \rightarrow \infty} \Pr[\mathcal{E}_n] = 1$ and with positive probability if $\liminf_{n \rightarrow \infty} \Pr[\mathcal{E}_n] > 0$. The following was first put forward in [7] and has become a folklore conjecture.

Satisfiability Threshold Conjecture For every $k \geq 2$, there exists a constant r_k such that for all $\epsilon > 0$,

- If $m \leq (r_k - \epsilon)n$ then w.h.p. $F_k(n, m)$ is satisfiable.
- If $m \geq (r_k + \epsilon)n$ then w.h.p. $F_k(n, m)$ is unsatisfiable.

¹While we adopt the $F_k(n, m)$ model throughout the paper, all the results we discuss hold in all standard models for random k -SAT, e.g. when clause replacement is not allowed and/or when each k -clause is formed by selecting k literals uniformly at random with replacement.

The satisfiability threshold conjecture has attracted attention in computer science, mathematics, and more recently in mathematical physics [17, 18, 19]. Below we briefly review the current state of knowledge.

For the linear-time solvable $k = 2$ case, the conjecture was settled independently, $r_2 = 1$, by Chvátal and Reed [7], Goerdts [13], and Fernandez de la Vega [9]; further, Bollobás et al. [4] recently determined the problem’s “scaling window”, $r_2 = 1 + \lambda n^{-1/3}$. For $k \geq 3$, not only the value of r_k remains unknown, but even its existence has not been established. However, a threshold sequence, $r_k(n)$, is known to exist by a theorem of Friedgut [10].

Theorem 1 ([10]) For every $k \geq 2$, there exists a sequence $r_k(n)$ such that for all $\epsilon > 0$,

- If $m \leq (r_k(n) - \epsilon)n$ then w.h.p. $F_k(n, m)$ is satisfiable.
- If $m \geq (r_k(n) + \epsilon)n$ then w.h.p. $F_k(n, m)$ is unsatisfiable.

An immediate corollary of Theorem 1 is that if $F_k(n, nr)$ is satisfiable with positive probability then, for all $\epsilon > 0$, $F_k(n, (r - \epsilon)n)$ is satisfiable w.h.p. While it remains open whether $r_k(n)$ converges (otherwise the conjecture would be resolved), to simplify notation in the rest of the paper we will write $r_k \geq \gamma$ to denote that $F_k(n, rn)$ is satisfiable w.h.p. for all $r < \gamma$ (analogously for $r_k \leq \gamma$).

Upper bounds for r_3 have all been based on probabilistic counting arguments. The best current bound $r_3 \leq 4.596$ is due to Janson, Stamatiou and Vamvakari [15], while recently Dubois, Boufkhad and Mandler [8] announced $r_3 \leq 4.506$. Lower bounds, on the other hand, have been algorithmic in nature. The first was given by Broder, Frieze and Upfal [5] who proved that w.h.p. the *pure literal* heuristic succeeds on $F_3(n, rn)$ for $r \leq 1.63$, but fails for $r \geq 1.7$.

Even before [5], Chao and Franco [6] had shown that algorithms UC and UCWM (to be defined in Section 2) find a satisfying truth assignment with positive probability for $r < 8/3$ and $r < 2.9$, respectively. Today, combining such results with the corollary of Theorem 1 gives lower bounds for r_3 . To improve over $r_3 \geq 1.63$, Frieze and Suen [11] considered a generalization of UC, called GUC, and showed

that it succeeds with positive probability iff $r < 3.003\dots$ To establish $r_3 \geq 3.003\dots$ they proved that a modified version of GUC, performing a very limited form of backtracking, succeeds w.h.p. for $r < 3.003\dots$

In a recent paper [2], the first author showed that all algorithms in [6, 7, 11] can be expressed in a common framework, and analyzed by employing differential equations. By introducing a new heuristic in this framework which “sets two variables at a time”, TT, he showed $r_3 > 3.145$.

In this paper, we consider *all* algorithms that fall within the framework of [2] (to be defined precisely in Section 2), collectively referring to them as “myopic” algorithms. A significant drawback of myopic algorithms hitherto analyzed is that rules for picking variable(s) to set, which we call *strategies*, and rules for setting them, which we call *policies*, are fixed throughout the algorithm’s execution. Here, we allow the strategy and policy to change continually in the course of the execution, taking advantage of the fact that different rules are better in different configurations.

Our main contribution is a methodology for determining an optimal policy for any given strategy.² Armed with this methodology, we first determine the optimal “one variable at a time” myopic algorithm (a class containing all algorithms in [6, 7, 11]) to establish $r_3 \geq 3.22$. By considering a strategy that can set two variables at a time, we then establish $r_3 \geq 3.26$, per the following theorem.

Theorem 2 $F_3(n, rn)$ is satisfiable w.h.p. for $r \leq 3.26$.

Our optimization of each policy is done by solving an instance of a problem we introduce, called “max-density multiple-choice knapsack”. The structure of knapsack solutions yields an elegant characterization of the optimal trade-off between the options available to a myopic algorithm.

Determining the optimal “two at a time” algorithm already poses significant computational challenges. One can consider even more complicated strategies, but at the expense of even greater complexity and – we speculate – with diminishing returns.

2 The Framework

All algorithms that have been analyzed on random formulae share two characteristics, both motivated by the need to maintain a simple form of conditional independence.

The first is that they are backtrack-free, *i.e.* once a variable is set to either 0 or 1, the setting is never reversed.³

²This paper is reminiscent of [21] in using mathematically non-trivial, computationally driven optimization to sharpen theoretical results.

³The backtracking version of GUC [11], which converts a positive-probability heuristic into a high-probability one, escapes this paradigm only superficially. In particular, w.h.p. no more than $\text{polylog}(n)$ value assignments are ever reversed, and any such reversal occurs for simple “local” reasons, leaving an unconditioned random formula. Similar backtracking can be retrofitted to all the algorithms we consider, raising their success rate from “positive” to “high”.

The second is that they are restricted in their variable-picking strategies and variable-setting policies. For myopic algorithms, the restriction is that at the beginning of each “round”, the algorithm’s “state” is captured only by the number, t , of unassigned variables and the number, m_i , of clauses of length $i = 1, 2, 3$: the residual formula F is uniformly random over all formulae with t variables and m_i i -clauses, which we write as $F \in \mathcal{F}(t; m_1, m_2, m_3)$. That is, for $i = 1, 2, 3$, the sets of remaining i -clauses form independent random i -SAT formulae $F_i(t, m_i(t))$. A myopic algorithm’s strategy and policy can depend arbitrarily on t and m_i , $i = 1, 2, 3$. We now make this concrete through an intuitive (but formalizable) model.

2.1 The Playing-Card Model

Imagine representing a k -SAT formula with a group of k cards for each k -clause, each card bearing the name of one literal. Originally all the cards are “face down”, so the literals are concealed. Assume that an intermediary knows which literal is on each card. To interact with the intermediary we are allowed to

- Point to a particular card, or
- Name a variable that has not yet been assigned a value.

In response, if the card we point to carries literal A , the intermediary reveals (turns face up) all the cards carrying A or \bar{A} . Similarly, if we name variable A , the intermediary reveals all the cards carrying A or \bar{A} .

Having chosen a variable, and depending on the revealed information, we may elect to choose another variable. In one “move” we are allowed to choose a constant number of variables before having to assign values to all of them.

Once we assign values to the variables chosen in a move, we remove all the cards corresponding to dissatisfied literals (converting an i -clause to an $(i - 1)$ -clause) and all the cards (some of them still face down) belonging to satisfied clauses. At the end of a move only “face-down” cards remain, containing only literals from unset variables.

We will insist that if there are any 1-clauses, also called unit clauses, a myopic algorithm should pick one of them (by pointing to its card) and satisfy it. This results in no loss of generality, per the following proposition.

Proposition 3 For any myopic algorithm \mathcal{A} , there exists a myopic algorithm \mathcal{B} which always satisfies a unit clause if one exists, and whose probability of satisfying a random formula in $\mathcal{F}(n; m_2, m_3)$ is at least as large as that of \mathcal{A} .

Let us summarize. A myopic algorithm operates on a formula F . If $m_1 = 0$, the algorithm makes a “free move”, sequentially picking up to a constant number of variables according to some “strategy” and then setting their values according to a “policy” (utilizing all the revealed information). This produces a smaller formula in which typically $m_1 \neq 0$. As long as $m_1 \neq 0$, the algorithm makes “forced moves”,

picking 1-clauses and satisfying them. When $m_1 = 0$ again, this “iteration” or “round”, of one free move and a series of forced moves, is over, yielding a formula F' .

Lemma 4 *Let $F \in \mathcal{F}(t; m_2, m_3)$ be a random formula. Apply one round of any myopic algorithm to F , producing formula F' . Conditional upon the iteration setting t_0 variables, and F' having (m'_2, m'_3) 2- and 3-clauses, F' is a random formula in $\mathcal{F}(t - t_0; m'_2, m'_3)$.*

2.2 Significance of 2-clauses

Since a myopic algorithm cannot backtrack, if it generates an empty clause (0-clause) during a forced move, then it fails. For the sake of our analysis, we will let the algorithm ignore 0-clauses and continue blithely on. The following lemma shows that if the 2-clause density remains bounded away from 1, then with positive probability no 0-clauses are generated. Conversely, since $r_2 = 1$, if the 2-clause density ever exceeds 1, then w.h.p. the 2-SAT subformula is unsatisfiable and the algorithm must fail.

Lemma 5 *Let A be any myopic algorithm. If there exist $\delta, \epsilon > 0$ and $t_f(n) \geq \epsilon n$ such that w.h.p. for all $t_f \leq t \leq n$, $m_2(t) < (1 - \delta)t$, then there exists $\rho = \rho(\delta, \epsilon) > 0$ such that $\Pr[m_0(t_f) + m_1(t_f) = 0] > \rho$.*

Proof. (Sketch.) When ρ_2 is bounded away from 1, the rate at which unit clauses are generated (per move) is also bounded away from 1. As a result, the number of unit clauses behaves like the queue size in a stable server system, implying $\sum_{t=n}^{t_f} m_1(t) = O(n)$. Since the probability that no 0-clause is generated during a forced move is proportional to $(1 - 1/t)^{m_1}$, and $t_f = \Omega(n)$, with positive probability no 0-clause is ever generated. \square

The next lemma shows that if for some r an algorithm can keep ρ_2 below 1 and bring $\rho_3 = m_3/t$ below $2/3$, this suffices to establish $r_3 \geq r$.

Lemma 6 *If for some myopic algorithm A , applied to $F_3(n, rn)$, there exist $\delta, \epsilon, \zeta > 0$ and $t_f(n) \geq \epsilon n$ such that w.h.p.: i) for all $t_f \leq t \leq n$, $m_2(t) < (1 - \delta)t$, and ii) $m_3(t_f) < (2/3 - \zeta)t_f$, then $r_3 \geq r$.*

Proof. (Sketch.) By Lemma 5, with positive probability, $m_1(t_f) + m_0(t_f) = 0$. Moreover, $m_2(t_f) < (1 - \delta)t_f$ and $m_3 < (2/3 - \zeta)t_f$. By a Theorem of [3], such mixtures of 2- and 3-clauses are satisfiable w.h.p., implying that $F_3(n, rn)$ is satisfiable with positive probability. Invoking the corollary of Theorem 1, $r_3 \geq r$. \square

Lemmata 5 and 6 show that it is vital to keep the 2-clause density below 1. All established algorithms make a forced move if unit clauses are present, but in free moves they take different approaches to keeping ρ_2 small. UC’s strategy is

to name a random unset variable, and its *policy* is to assign it 0 or 1 at random. UCWM’s strategy is also to name a random variable, but its policy is to set it so as to minimize the number of unsatisfied 3-clauses. GUC points to a random card in a random clause of shortest length, and sets its literal to 1. TT points to both cards in a random 2-clause, and sets their variables so as to simultaneously satisfy the clause and minimize the number of unsatisfied 3-clauses; if no 2-clauses exist, it names a random variable and assigns it 0 or 1 at random.

Outline Our method for determining the optimal policy can be applied to any given strategy. We demonstrate the method in Sections 3–5 for a particular strategy, SC, which will turn out to be the optimal “one variable at a time” strategy. Section 6.3 proves that this strategy and policy satisfy $F_3(n, rn)$ with positive probability if $r < r_3^* \approx 3.22$. In Section 6.4 we prove the optimality of the policy derived by our method: for SC, no policy can satisfy $F_3(n, rn)$ with positive probability for $r > r_3^*$. In Section 7 we consider alternative strategies, including ones that set more than one variable at a time.

3 The shortest clause strategy (SC)

We will show how to develop an optimal policy for the “shortest clause” strategy, or SC, a flavor of which (GUC) was studied in [11]. In view of Lemma 6, we can restrict our attention to cases $\rho_2 < 1$ (or else the residual formula is unsatisfiable w.h.p., and what we do is irrelevant). It will also suffice to stop the algorithm before $t < .1n$, and thus we may assume $t = \Omega(n)$. Finally, starting from $m_2 = 0$, the first ϵn moves will change the clause densities only a little, but will build up a supply of 2-clauses that will never run out; for this reason we can assume that our strategy can always pick a 2-clause. These assertions will be borne out by the analysis to come.

SC operates as follows. In a free move, SC select a 2-clause at random, and chooses at random one of its two literals, A . (W.l.o.g., it is convenient to imagine A to be a positive occurrence of a variable.) Count positive and negative occurrences of A in 3-clauses and 2-clauses, including the selected 2-clause, to define a 4-vector $(A_2^+, A_2^-, A_3^+, A_3^-)$. Set A to either 0 or 1 according to some *policy*, about which we will say more shortly. While there are any unit clauses, choose one at random, and set its literal $A = 1$.

In GUC, the *policy* in free moves is always to set $A = 1$. We will allow far more flexible policies that can depend arbitrarily on $Q \equiv (t, m_2, m_3)$. Thus, a policy is a family of functions $\sigma_Q : \mathbb{Z}^4 \rightarrow \{0, 1\}$, and the algorithm sets $A = \sigma_Q(A_2^+, A_2^-, A_3^+, A_3^-)$. To recapitulate, given a family of functions σ_Q , we consider repeated iterations (rounds) of:

- Choose a 2-clause at random, and from it choose a literal A at random. Set $A = \sigma_Q(A_2^+, A_2^-, A_3^+, A_3^-)$.
[Free move]
- While there are unit clauses, choose one at random, and set its literal $A = 1$.
[Forced moves]

3.1 Motivating the optimal policy

In the rest of this section and Sections 4–6, we will (i) define a particular policy σ^* , and (ii) analyze the performance of the shortest-clause strategy using σ^* . While the proof that σ^* is optimal will have to wait for Section 6.4, we will start by intuitively motivating the construction of σ^* .

As a rule of thumb, it is good to satisfy “as many clauses as possible.” Hence, if $A_2^+ \geq A_2^-$ and $A_3^+ \geq A_3^-$ then we should set $A = 1$. (This produces a random formula with fewer 2-clauses and fewer 1-clauses than setting $A = 0$ would; such a formula is more likely to be satisfiable, and we would expect that it is also more likely to be satisfied by our algorithm.) However, what if setting $A = 1$ satisfies more 3-clauses, while setting $A = 0$ satisfies more 2-clauses? How should we balance these considerations?

It will turn out that the appropriate parameterization is the 2-clause and 3-clause densities $\rho_2 = m_2/t$ and $\rho_3 = m_3/t$. The policies we will focus on all yield random processes that are macroscopically predictable: when an algorithm is run on a random formula, w.h.p., the observed values (ρ_2, ρ_3) lie almost exactly on a fixed “trajectory” in (ρ_2, ρ_3) -space that depends only on the policy and the initial value of (ρ_2, ρ_3) . In Figure 1, for example, we see the trajectory for the shortest-clause strategy with policy σ^* , starting from $(\rho_2, \rho_3) = (0, 3.22)$, and terminating when $\rho_3 = 2/3$ and $\rho_2 < 1$.

Intuitively, policies giving flatter trajectories should be better, for we would then be able to reach $(\rho_2, \rho_3) = (2/3, 1)$ starting from a larger value of ρ_3 (and $\rho_2 = 0$). To put it another way, as the algorithm “moves” along the trajectory, at a given value of ρ_3 , it would be better to have a smaller value of ρ_2 . (This would make the formula more likely to be satisfiable, and so perhaps more likely to be satisfied by our algorithm). Thus, it is natural to consider the policy maximizing the trajectory’s slope at each point. (Note that as ρ_3 is decreasing, and ρ_2 increasing, the slopes are negative).

Technically speaking, the following sequence of calculations simply defines a number of quantities, in terms of which we will define our policy σ^* . While we will only later prove the connection between these quantities and the (ρ_2, ρ_3) -space behavior described above, the calculations are more easily understood with these notions in mind.

3.2 Defining the optimal policy

Let $P(\lambda; i)$ denote the probability that a Poisson random variable with mean λ takes the value i . Letting $\rho_2 =$

m_2/t and $\rho_3 = m_3/t$, the probability of observing a tuple $(A_2^+, A_2^-, A_3^+, A_3^-)$ in a free move of SC is, up to $o(1)$,

$$\Pr(A_2^+, A_2^-, A_3^+, A_3^-) = \Pr\left(\frac{3}{2}\rho_3; A_3^+\right) \Pr\left(\frac{3}{2}\rho_3; A_3^-\right) \Pr(\rho_2; A_2^+ - 1) \Pr(\rho_2; A_2^-). \quad (1)$$

(It is $A_2^+ - 1$ because the appearance of A in the chosen 2-clause comes for free.)

Given a policy σ_Q and a tuple \vec{A} , define $U_3(\vec{A})$ to be the number of 3-clauses unsatisfied by setting $A = \sigma_Q(\vec{A})$, and define $U_2(\vec{A})$ likewise. Thus, if $A = 1$, $U_3 = A_3^-$ and $U_2 = A_2^-$, while if $A = 0$, $U_3 = A_3^+$ and $U_2 = A_2^+$.

Lemma 7 Starting with $F \in \mathcal{F}(t; m_2, m_3)$, a single round of SC produces a random formula $F \in \mathcal{F}(t + \Delta t; m_2 + \Delta m_2, m_3 + \Delta m_3)$ where, up to $o(1)$ terms,

$$\mathbb{E}(\Delta t) = -1 - \frac{\mathbb{E}(U_2)}{1 - \rho_2}, \quad (2)$$

$$\begin{aligned} \mathbb{E}(\Delta m_3) &= -3\rho_3 - \frac{\mathbb{E}(U_2)}{1 - \rho_2} (3\rho_3) \\ &= 3\rho_3 \mathbb{E}(\Delta t), \end{aligned} \quad (3)$$

$$\begin{aligned} \mathbb{E}(\Delta m_2) &= \mathbb{E}(U_3) - 2\rho_2 + \frac{\mathbb{E}(U_2)}{1 - \rho_2} \left(\frac{3}{2}\rho_3 - 2\rho_2\right) \\ &= \left(2\rho_2 - \frac{3}{2}\rho_3\right) \mathbb{E}(\Delta t) + \left(\mathbb{E}(U_3) - \frac{3}{2}\rho_3 - 1\right). \end{aligned} \quad (4)$$

Proof. In one round, conditional upon a tuple \vec{A} for the free move, and taking expectations of similar counts over the ensuing sequence of forced moves, up to $o(1)$ terms,

$$\mathbb{E}(\Delta t | \vec{A}) = -1 - \frac{U_2(\vec{A})}{1 - \rho_2} \quad (5)$$

$$\mathbb{E}(\Delta m_3 | \vec{A}) = (-A_3^+ - A_3^-) + \frac{U_2(\vec{A})}{1 - \rho_2} (-3\rho_3) \quad (6)$$

$$\begin{aligned} \mathbb{E}(\Delta m_2 | \vec{A}) &= (U_3(\vec{A}) - A_2^+ - A_2^-) \\ &\quad + \frac{U_2(\vec{A})}{1 - \rho_2} \left(\frac{3}{2}\rho_3 - 2\rho_2\right), \end{aligned} \quad (7)$$

where in each case we write the contribution of the free move before that of the $U_2/(1 - \rho_2)$ forced moves. Equations (2)–(4) represent the expectations of the quantities in (5)–(7) under the distribution of \vec{A} given by (1). \square

Definition 8 Given any policy σ , define

$$\lambda_\sigma(\rho_2, \rho_3) = \frac{\mathbb{E}(\Delta m_2) - \rho_2 \mathbb{E}(\Delta t)}{\mathbb{E}(\Delta m_3) - \rho_3 \mathbb{E}(\Delta t)} \quad (8)$$

Let $\lambda^*(\rho_2, \rho_3) = \max_\sigma \lambda_\sigma(\rho_2, \rho_3)$, and let $\sigma^*(\rho_2, \rho_3)$ be a policy achieving the maximum.

These σ^* will be our optimal policies!

Intuitively, $\lambda_\sigma(\rho_2, \rho_3)$ is the ratio of a “typical” change in ρ_2 to a typical change in ρ_3 , each change defined as $\Delta\rho_i \doteq \frac{m_i - \Delta m_i}{t - \Delta t} - \frac{m_i}{t}$, $i = 2, 3$. To see why $\Delta\rho_2/\Delta\rho_3$ takes the form appearing in (8), observe that $\frac{y + \Delta y}{x + \Delta x} - \frac{y}{x} \approx \frac{1}{x}(\Delta y - \frac{y}{x}\Delta x)$; take the ratio of the substitutions $y = m_2$, $x = t$, and $y = m_3$, $x = t$.

Remember that in equations (5)–(7), U_3 and U_2 are functions of \vec{A} and of the *policy* – of whether $A = \sigma_Q(\vec{A})$ is 0 or 1. Rewrite the RHS of (8) as

$$\frac{\sum_{\vec{A}} \Pr(\vec{A}) \left[\mathbb{E}(\Delta m_2 | \vec{A}, \sigma(\vec{A})) - \rho_2 \mathbb{E}(\Delta t | \vec{A}, \sigma(\vec{A})) \right]}{\sum_{\vec{A}} \Pr(\vec{A}) \left[\mathbb{E}(\Delta m_3 | \vec{A}, \sigma(\vec{A})) - \rho_3 \mathbb{E}(\Delta t | \vec{A}, \sigma(\vec{A})) \right]} \quad (9)$$

and then rewrite (8) as $\lambda_\sigma(\rho_2, \rho_3) = \frac{\sum_{\vec{A}} y_{\vec{A}}^{\sigma(\vec{A})}}{\sum_{\vec{A}} x_{\vec{A}}^{\sigma(\vec{A})}}$.

To compute σ^* we must choose the values $\sigma(\vec{A}) \in \{0, 1\}$ to maximize this ratio, *i.e.* (simplifying indices) to maximize $\sum_i y_i^j / \sum_i x_i^j$, by choosing the best j for each i .

4. A knapsack problem

The following may be thought of as a variant of the NP-complete “Knapsack” problem [12, problem MP9].

MAX-DENSITY MULTIPLE-CHOICE KNAPSACK

Instance: A set of pairs $\{(x_i^j, y_i^j)\}$, for $i = 1, \dots, n$ and $j = 1, \dots, k$. The x_i^j satisfy the condition that for any values $j = \sigma(i)$, $\sum_i x_i^{\sigma(i)} > 0$. Also given is a value λ .

Decision: Does there exist a function $\sigma(i)$ such that $\sum_i y_i^{\sigma(i)} / \sum_i x_i^{\sigma(i)} \geq \lambda$?

Optimization: Find σ to maximize $\sum_i y_i^{\sigma(i)} / \sum_i x_i^{\sigma(i)}$.

(We will see in Section 5 that the knapsack instances defined by (8) indeed satisfy the positivity condition above.) Without the positivity condition on the x ’s, the decision problem, with integer inputs, is NP-complete. The proof is a simple reduction from Partition [12, problem SP12]. However, with the stated constraint on the x ’s, the situation is much better.

Theorem 9 *The max-density multiple-choice knapsack optimization problem can be solved in time $O(nk \log(nk))$.*

The proof of Theorem 9 constitutes the rest of this section.

Note that the problem is not trivial: choosing for each i the pair with largest $y_i^{\sigma(i)} / x_i^{\sigma(i)}$ does not work. For example, for some i , all pairs might have above-average slope, and a pair with both x and y large can swing the average more than a pair having larger ratio but smaller values:

To solve knapsack, let us first solve a different problem: for a given “input” slope λ , maximize the “height” $h = \sum_i y_i^{\sigma(i)} - \lambda \sum_i x_i^{\sigma(i)}$. This is trivial: for each i , independently, choose $\sigma(i)$ to maximize $y_i^{\sigma(i)} - \lambda x_i^{\sigma(i)}$. Note that this takes time nk and let $\lambda' = \sum_i y_i^{\sigma(i)} / \sum_i x_i^{\sigma(i)}$.

Now, if $h > 0$, then $\lambda' > \lambda$, and thus for the optimal slope also, $\lambda^* > \lambda$. (This is where we use the hypothesis that $\sum x_i^{\sigma(i)} > 0$; were we in the left half-plane, $h > 0$ would correspond to a *smaller* slope.) Conversely, if $\lambda^* > \lambda$, then the corresponding, optimal policy has $\sum y_i^{\sigma(i)} = \lambda^* \sum x_i^{\sigma(i)} > \lambda \sum x_i^{\sigma(i)}$, and thus $h > 0$. By the same token, $h = 0$ iff $\lambda^* = \lambda$. In short, the “height maximization” algorithm above allows us to determine if a given λ is larger than, equal to, or smaller than the optimal slope, enabling a binary search strategy on λ . To show convergence in finite time, we still must show that there are only so many relevant values of λ to choose between.

For any set Q of points $(x_i^1, y_i^1), \dots, (x_i^k, y_i^k)$, for any λ , the point $(x_i^{\sigma(i)}, y_i^{\sigma(i)})$ that maximizes $y - \lambda x$ lies on the upper portion of the convex hull of Q . If this “upper hull” contains $k' \leq k$ points, the particular point that gives the maximum depends on the value of λ compared with the $k' - 1$ slopes between adjacent points on the hull. A convex hull on k points can be constructed in time $O(k \log k)$ with “Graham’s scan” [14]. Over all n k -way choices, then, there are at most $n(k - 1)$ critical values of λ .

The full algorithm, then, is: For each of the n sets, compute the convex hull (time $O(nk \log k)$). Compute the corresponding $n(k - 1)$ or fewer critical slopes, and sort them (time $O(nk \log(nk))$). Using the height-maximization subroutine, run binary search on this list; this requires $O(\log(nk))$ iterations, each taking time $O(nk)$ (total time $O(nk \log(nk))$). Once λ is found to lie between two of the critical values, running the subroutine on any value between them produces the optimal policy, and with it the optimal slope λ^* . Given λ^* , σ^* is characterized by:

*for each i , independently, choose the pair
maximizing $y_i^{\sigma(i)} - \lambda^* x_i^{\sigma(i)}$.*

5 The structure of optimal policies

The structure of optimal knapsack solutions gives an elegant characterization of optimal policies.

First, let us verify that knapsack instances defined by (8) satisfy the positivity condition. For *any* myopic algorithm, independently of the strategy used, in every move the expected number of 3-clauses containing the variable set is at least $3\rho_3$: pointing to a card in a 3-clause raises this expectation to $3\rho_3 + 1$, while otherwise we get $3\rho_3$ since the variable choice is independent of the 3-clauses. Then, $\mathbb{E}(\Delta\rho_3)/\mathbb{E}(\Delta t) \geq 3\rho_3$, and $\mathbb{E}(\Delta m_3) - \rho_3 \mathbb{E}(\Delta t) < 0$; negating all quantities gives a valid instance.

For SC in particular, $\mathbb{E}(\Delta m_3) - \rho_3 \mathbb{E}(\Delta t) = 2\rho_3 \mathbb{E}(\Delta t)$; thus, by (8), maximizing $\lambda_\sigma(\rho_2, \rho_3)$ amounts to maximizing $(-\mathbb{E}(\Delta m_2))/(-\mathbb{E}(\Delta t))$. From the characterization of optimal knapsack solutions, it follows that for each 4-tuple \vec{A} the optimal choice is the one minimizing

$$\mathbb{E}(\Delta m_2 | \vec{A}, \sigma(\vec{A})) - \lambda^* \mathbb{E}(\Delta t | \vec{A}, \sigma(\vec{A})), \quad (10)$$

for some constant λ^* . Substituting (5), (7) into (10), this implies that we set $A = 1$ iff $A_3^+ - A_3^- \geq \theta(A_2^- - A_2^+)$, where $\theta = (\frac{3}{2}\rho_3 - 2\rho_2 + \lambda^*)/(1 - \rho_2)$.

This is quite telling. First, the decision depends only on $A_3^+ - A_3^-$ and $A_2^+ - A_2^-$, rather than the full 4-tuple $(A_2^+, A_2^-, A_3^+, A_3^-)$. It confirms our earlier intuition that if a setting maximizes both the number of 2- and 3-clauses satisfied then it should be preferred. If no such setting exists, the optimal policy trades off generating fewer 2-clauses now (measured by $A_3^+ - A_3^-$) against having more possibilities to do so in the future (measured by $A_2^- - A_2^+$; for each unsatisfied 2-clause we expect $1/(1 - \rho_2)$ forced moves). The “exchange rate”, set by the constant θ , weights $A_2^- - A_2^+$. Recalling that $\mathbb{E}(\Delta m_2)/(-\mathbb{E}(\Delta t))$ is $3\rho_3 - 2\rho_2$ in forced moves but $-\lambda^*$ overall, θ is proportional to: i) the gain, $(\frac{3}{2}\rho_3 - 2\rho_2) - (-\lambda^*)$, afforded by free moves in suppressing 2-clauses, and ii) the scarcity, $1/(1 - \rho_2)$, of free moves.

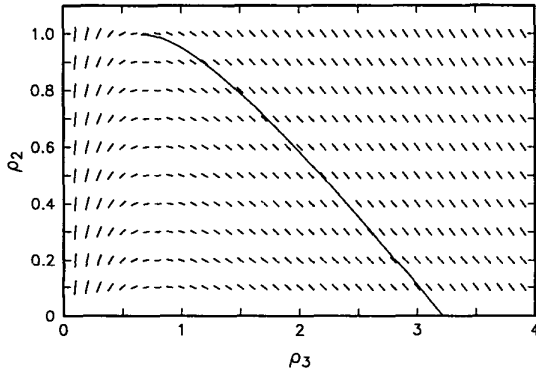


Figure 1. Optimal gradients $d\rho_2/d\rho_3$ for “one variable at a time”, and the trajectory starting from $(3.22, 0)$ and stopping when $\rho_3 < 2/3$.

6 Analysis of the SC algorithm

In this section we will first analyze the knapsack-derived policy σ^* to show that – coupled with the strategy (SC) of picking a random literal from a random shortest clause – it satisfies random formulas of density 3.22 or less. In section 6.4 we will re-use the same fundamentals to show that σ^* is an optimal policy. In reading this section, it is helpful to keep Figure 1 in mind.

6.1 Properties of the algorithm and policy

Lemma 10 For any fixed policy σ (not depending on t , m_2 , or m_3), the slope $\lambda_\sigma(\rho_2, \rho_3)$ is Lipschitz continuous. Also, $\lambda^*(\rho_2, \rho_3)$ (per Definition 8) is Lipschitz continuous.

Proof. We begin with the fixed policy case. To show Lipschitz continuity of $\mathbb{E}((1 - \rho_2)\Delta t) = \sum_{\vec{A}} \Pr(\vec{A})\mathbb{E}((1 - \rho_2)\Delta m_2 \mid \vec{A}, \sigma(\vec{A}))$ is essentially to show Lipschitz continuity of $\mathbb{E}U_2$; as U_2 is Poisson this is elementary. The

same goes for $\mathbb{E}((1 - \rho_2)\Delta m_2)$ and $\mathbb{E}((1 - \rho_2)\Delta m_3)$. Then, Lipschitz continuity of products of bounded Lipschitz continuous functions, and of quotients whose denominator is bounded away from 0, shows Lipschitz continuity of $\lambda(\rho_2, \rho_3) = \frac{(1 - \rho_2)\mathbb{E}(\Delta m_2) - (1 - \rho_2)\rho_2\mathbb{E}(\Delta t)}{(1 - \rho_2)\mathbb{E}(\Delta m_3) - (1 - \rho_2)\rho_3\mathbb{E}(\Delta t)}$.

Now, let $\mathbf{x} = (\rho_2, \rho_3)$. Let two points \mathbf{x}_1 and \mathbf{x}_2 , have optimal policies σ_1 and σ_2 . Taking advantage of Lipschitz continuity of any fixed policy, with constant K , $\lambda^*(\mathbf{x}_1) = \lambda_{\sigma_1}(\mathbf{x}_1) \leq \lambda_{\sigma_1}(\mathbf{x}_2) + K|\mathbf{x}_1 - \mathbf{x}_2| \leq \lambda^*(\mathbf{x}_2) + K|\mathbf{x}_1 - \mathbf{x}_2|$, so $\lambda^*(\mathbf{x}_1) - \lambda^*(\mathbf{x}_2) \leq K|\mathbf{x}_1 - \mathbf{x}_2|$. From symmetry, $|\lambda^*(\mathbf{x}_1) - \lambda^*(\mathbf{x}_2)| \leq K|\mathbf{x}_1 - \mathbf{x}_2|$. \square

Surprisingly, although the slope $\lambda^*(\rho_2, \rho_3)$ is Lipschitz continuous, the numerator and denominator defining it are not. Even an infinitesimal change to (ρ_2, ρ_3) may change the policy choice for some case \vec{A} ; in one move such a tuple is observed with constant probability, and thus $\mathbb{E}(\Delta t)$, $\mathbb{E}(\Delta m_2)$, and $\mathbb{E}(\Delta m_3)$ all change by $\Theta(1)$. We will return to this point.

In the following, where $w(n) \rightarrow \infty$, we use “with overwhelming probability”, or “w.o.p.”, to refer to probability $1 - \exp(-\Theta(w(n)))$.

Lemma 11 Let $w(n)$ be any function satisfying $w(n) \rightarrow \infty$ and $w(n) = o(n)$. Given any densities (ρ_2, ρ_3) with $\rho_2 < 1$, let $m_2(n)$ and $m_3(n)$ be any sequences with $m_2/n \rightarrow \rho_2$, and $m_3/n \rightarrow \rho_3$. Apply strategy SC with policy σ^* to a random $F \in \mathcal{F}(n; m_2(n), m_3(n))$, for $w(n)$ rounds, to yield $F' \in \mathcal{F}(n'; m'_2, m'_3)$.

Then w.o.p., $\Delta n \doteq n' - n = o(n)$, $\Delta m_2 \doteq m'_2 - m_2 = o(n)$, and $\Delta m_3 \doteq m'_3 - m_3 = o(n)$. Furthermore, $\Delta\rho_2/\Delta\rho_3 \doteq (m'_2/n' - m_2/n)/(m'_3/n' - m_3/n) = \lambda^* + o(1)$, where λ^* is as per Definition 8.

Proof. Let $X(\lambda)$ denote a Poisson random variable with mean λ . In one free move, the number of 1-clauses created is $M_1 \leq 1 + X(\rho_2)$ (if A is set to 1, M_1 is only $X(\rho_2)$). In each forced move, exactly one 1-clause is consumed, while another $X(\rho_2)$ are added, so the number of forced moves before the “queue” empties is the first time τ_1 at which $M_1 - \tau_1 + X(\tau_1\rho_2) = 0$, where $X(t)$ is now to be thought of as a single Poisson process. At time τ_1 , a new free move is made, inducing τ_2 new forced moves, and so forth. The total number of forced moves, $\tau = \sum_1^{w(n)} \tau_i$, is the smallest value for which $(\sum_i M_i) - \tau + X(\tau\rho_2) = 0$.

$M = \sum M_i$ stochastically dominates $X(w(n)\rho_2)$, and w.o.p. $M = \Theta(w(n))$. It is a standard result in queuing theory that $\mathbb{E}(\tau) = M/(1 - \rho_2)$. If at time t the queue length $M - t + X(t\rho_2)$ is 0 (or negative), then the queue first emptied at $\tau \leq t$; thus $\Pr[\tau > 2\mathbb{E}(\tau)] \leq \Pr[X(M(2/(1 - \rho_2))) > M(2/(1 - \rho_2) - 1)]$. For fixed ρ_2 , this means that a Poisson r.v. with mean $\Theta(M) = \Theta(w)$ exceeds its expectation by a constant factor, which happens with probability exponentially small in $w(n)$. Thus w.o.p. the number of free and forced moves is $\Theta(w(n))$, and the

number of variables “touched”, Poisson in the number of moves, is also $\Theta(w(n))$ w.o.p. In the course of the round, then, n , m_2 , and m_3 change by factors $1 + o(1)$, and the same follows for ρ_2 and ρ_3 . This justifies our neglect to account for change in ρ_2 and ρ_3 as the round progresses.

Next, consider the knapsack instances, by which σ^* is defined, generated during the round. Each case $\vec{A} = (A_2^+, A_2^-, A_3^+, A_3^-)$ happens w.p. a continuous function of ρ_2 and ρ_3 , and changes by factors $1 + o(1)$. By Lemma 10, the optimal “slope” λ^* for these knapsacks also changes only by $1 + o(1)$, and the only knapsack decisions changing within the round are those for which $y^0 - \lambda^* x^0 = y^1 - \lambda^* x^1 + o(1)$. By definition of λ^* and σ^* , in one round, with ρ_2 and ρ_3 artificially “frozen”, $\mathbb{E}(y_{\vec{A}}^{\sigma^*(\vec{A})} - \lambda^* x_{\vec{A}}^{\sigma^*(\vec{A})}) = 0$. In $\Theta(w)$ moves every case is observed almost exactly as many times as expected; over w rounds then, and even allowing σ to make arbitrary choices in the borderline cases, w.o.p. $\sum (y^\sigma - \lambda^* x^\sigma) = o(w)$. At the same time, $\sum x^\sigma = \sum (\Delta m_3 - \rho_3 \Delta t) = \Theta(w)$, w.o.p. Thus w.o.p. $(\sum y) / (\sum x) = \Delta \rho_2 / \Delta \rho_3 = \lambda + o(1)$. \square

In line with the remark after Lemma 10 that λ^* is Lipschitz continuous but its defining numerator and denominator are not, Lemma 11’s $\Delta \rho_2 / \Delta \rho_3$ is predictable almost exactly, but the vector $(\Delta \rho_2, \Delta \rho_3)$ is not. This makes it impossible to apply the “packaged” differential-equation analyses in [16, 22], requiring the following lemma instead.

6.2 Differential equations for random processes

Lemma 12 *Given are*

- (1) *an n -indexed sequence of Markov chains, where each such chain (suppressing the index n) is represented as $Z(0), Z(1), \dots \in S$,*
- (2) *projections $x : S \rightarrow \mathbb{R}$ and $y : S \rightarrow \mathbb{R}$, and*
- (3) *a “line field” (slope function) $\lambda(x, y)$, which, over some bounded, connected open set D , is bounded and Lipschitz continuous with Lipschitz constant K , and therefore for each (x_0, y_0) has a unique “flow” $f : \mathbb{R} \rightarrow \mathbb{R}$ satisfying $f(x_0) = y_0$ and $df(x)/dx = \lambda(x, f(x))$.*

Suppose that, starting from a fixed value $Z(0) = Z_0$, there exists a sequence of times $i_0 = 0, i_1, \dots, i_m$ ($m = m(n)$, and, like “stopping rules”, i_j may depend on $Z(0), \dots, Z(i_j)$) such that w.h.p., for every j , in the period from i_j to i_{j+1} , with $\Delta x = x(Z(i_{j+1})) - x(Z(i_j))$ and $\Delta y = y(Z(i_{j+1})) - y(Z(i_j))$:

- (4) $\Delta x > 0$, and $|\Delta x|, |\Delta y|$ are both $o(1)$ (as $n \rightarrow \infty$),
- (5) $|\frac{\Delta y}{\Delta x} - \lambda(x(Z(i_j)), y(Z(i_j)))| = o(1)$, and
- (6) $(x(Z(i_{j+1})), y(Z(i_{j+1}))) \in D$.

Then, for all $\varepsilon > 0$, for all n sufficiently large, w.h.p., for all j , $|y(Z(i_j)) - f(x(Z(i_j)))| < \varepsilon$.

Proof. For clarity, we will write x_j in lieu of $x(Z(i_j))$, and so forth. We shall only prove a high-probability lower bound, namely that $\Pr(y_j > f(x_j) + \varepsilon) = o(1)$. An identically obtained upper bound completes the argument.

Consider a line field defined by $\lambda^+(x, y) = \lambda(x, y) + \delta$ for some small constant $\delta > 0$. The corresponding flows f^+ satisfy $f^+(x) > f(x)$ for all $x > x_0$, and, for any $\varepsilon > 0$, for δ sufficiently small, for any $(x, y) \in D$, $|f^+(x) - f(x)| < \varepsilon$. (See e.g. [20, Lemma 4.8].) *I.e.*, each flow f^+ lies above the corresponding f , but never by more than ε .

Consider the flow f^+ through the point (x_j, y_j) . Note now that by the Lipschitz continuity of λ , $f^+(x_{j+1}) \geq f^+(x_j) + [\lambda(x_j, y_j) + \delta - (K|\Delta x| + K|\Delta y|)]\Delta x$. Choose the index n sufficiently large that $|\Delta x|$ and $|\Delta y|$ are sufficiently small that $K|\Delta x| + K|\Delta y|$ is at most $\delta/2$, and thus $f^+(x_{j+1}) \geq f^+(x_j) + [\lambda(x_j, y_j) + \delta/2]\Delta x$.

On the other hand, by hypothesis (5), almost surely, $y_{j+1} = y_j + [\lambda(x_j, y_j) + o(1)]\Delta x$. Choose n sufficiently large that in this inequality, $o(1) < \delta/3$, so that $y_{j+1} \leq y_j + [\lambda(x_j, y_j) + \delta/3]\Delta x$.

From the previous two paragraphs, $y_{j+1} \leq f^+(x_{j+1})$ almost surely, where $f^+(x)$ is the δ -perturbed flow through (x_j, y_j) . “Re-start” both the Markov chain and f^+ from (x_{j+1}, y_{j+1}) , and note that (x_{j+2}, y_{j+2}) lies below the f^+ flow through (x_{j+1}, y_{j+1}) which in turn lies below the f^+ flow through (x_j, y_j) . Repeat, to conclude that, where f^+ is the flow through (x_0, y_0) , w.h.p., for all j , $y_j \leq f^+(x_j) \leq f(x_j) + \varepsilon$; the failure probability is no more than the sums of the failure probabilities in each step which by hypothesis is $o(1)$. \square

6.3 Conclusions for the algorithm

Lemmata 10 and 11 show that for the shortest-clause strategy with policy σ^* , the hypotheses of Lemma 12 are obeyed (modulo a negation of x), using $w(n) = \sqrt{n}$ and times $i_j = j w(n)$. We therefore have the following.

Theorem 13 *With high probability, in an execution of the SC strategy with policy σ^* on $F \in \mathcal{F}(n; m_2, m_3)$, the sequence of values of $(m_2/t, m_3/t)$ almost exactly follows the flows defined by the line field λ^* .*

In the line field λ^* , let $\rho_3^* \approx 3.22$ be the point where the flow through $(\rho_2, \rho_3) = (1, 2/3)$ intersects the axis $\rho_2 = 0$. (Section 6.5 discusses how we computed the value 3.22.) Then for any $r < \rho_3^*$, the flow through $(r, 0)$ passes through a point $(2/3, \rho_2)$ with $\rho_2 < 1$; let $\varepsilon(r) = 1 - \rho_2$. The flow through $(r, 0)$ is bounded away from 1 (by ε), so for sufficiently large n , w.h.p. an execution of the algorithm on $F \in \mathcal{F}(n, rn)$ also has densities which stay bounded away from $\rho_2 = 1$ (say by $\varepsilon/2$), and stay close to the flow (say by $\varepsilon/2$ again), and therefore when ρ_3 crosses $2/3$ has $\rho_2 < 1 - \varepsilon/2$. Such a formula is satisfiable w.h.p. If no 0-clauses were generated – that is, with positive probability

– the original formula was also satisfiable; the corollary of Friedgut’s theorem then implies satisfiability w.h.p.

6.4 Optimality of the algorithm

Arguments similar to the foregoing suffice to prove asymptotic optimality of our policy for the SC strategy. First, in analogy with Lemma 11, we have the following.

Lemma 14 *Let $w(n)$ be any function satisfying $w(n) \rightarrow \infty$ and $w(n) = o(n)$. Given any densities (ρ_2, ρ_3) with $\rho_2 < 1$, let $m_2(n)$ and $m_3(n)$ be any sequences with $m_2/n \rightarrow \rho_2$, and $m_3/n \rightarrow \rho_3$. For any policy σ , apply strategy SC with policy σ to a random $F \in \mathcal{F}(n; m_2(n), m_3(n))$, for $w(n)$ rounds, to yield $F' \in \mathcal{F}(n'; m'_2, m'_3)$.*

Then w.o.p., $\Delta n \doteq n' - n = o(n)$, $\Delta m_2 \doteq m'_2 - m_2 = o(n)$, and $\Delta m_3 \doteq m'_3 - m_3 = o(n)$. Furthermore, $\Delta \rho_2 / \Delta \rho_3 \doteq (m'_2/n' - m_2/n) / (m'_3/n' - m_3/n) \leq \lambda^ + o(1)$, where λ^* is as per Definition 8.*

Proof. All but the bound on $\Delta \rho_2 / \Delta \rho_3$ is proved just as in Lemma 11.

With respect to the distance $\sum_i \sum_j [(x_i^j - x_i'^j)^2 + (y_i^j - y_i'^j)^2]$ between two knapsack instances, the optimal slope λ^* is continuous. Consider one knapsack instance to be the “nominal” one, per (9), and the second to be similar but with $\Pr(\bar{A})$ replaced by the actual fraction of the time that the event \bar{A} occurred in the w rounds. For case probabilities p_i and “payoffs” bounded by w_i , the typical distance is $\frac{1}{n} \sum_{i=1}^{\infty} p_i(1 - p_i)w_i$; in our case where roughly $p_i = P(\lambda; w_i)$, this is $O(1/n)$, and w.h.p. the “sample” instance’s slope λ is $\lambda^*(\rho_2, \rho_3) + o(1)$. \square

Second, we have a result analogous to Lemma 12.

Lemma 15 *Under the hypotheses of Lemma 12, with condition (5) replaced by $\Delta y / \Delta x < \lambda(x(Z(i_j)), y(Z(i_j)))$, for all n sufficiently large, w.h.p., for all j , $y(Z(i_j)) < f(x(Z(i_j)))$.*

Proof. (Sketch.) The proof is like one direction of that of Lemma 12. Roughly, at each time i_{j+1} , w.h.p., the Markov process lies below the flow line on which it started at time i_j . Because the flow lines do not cross, the Markov process ends below the flow line on which it began. \square

Combining these two lemmata, we reach a conclusion analogous to that of Theorem 13.

Theorem 16 *For any $\varepsilon > 0$, with high probability, in an execution of the SC strategy with an arbitrary policy σ on $F \in \mathcal{F}(n; m_2, m_3)$, the sequence of values of $(m_2/t, m_3/t)$ lies above the flows defined by the line field $\lambda^* - \varepsilon$.*

(The switch from “below” in the Lemma to “above” in the Theorem is cosmetic, and due to the fact that our Δx is negative, where the Lemma calls for it to be positive.)

Before tying this up, we remark that the line field $\lambda^*(\rho_2, \rho_3)$, which through equations (2)–(4) was only properly defined for $\rho_2 \in [0, 1]$, can be extended smoothly to all ρ_2 ; the proportion of free moves approaches 0 as ρ_2 approaches 1, and for $\rho_2 > 1$ the proportion remains 0. By the same token, the algorithm runs perfectly well even when $\rho_2 > 1$, and its densities ρ_2 and ρ_3 continue to be predictable from the flows of λ^* ; the only difference (but a vital one) is that for $\rho_2 > 1$ the number of 1-clauses will increase indefinitely, and 0-clauses will be produced w.h.p.

As before, let $\rho_3^* \approx 3.22$ be the point where the flow through $(\rho_2, \rho_3) = (1, 2/3)$ intersects the axis $\rho_2 = 0$. Any flow above this one crosses the line $\rho_2 = 1$. Then for any $r > \rho_3^*$, the λ^* -flow through $(r, 0)$ crosses $\rho_2 = 1$. By Lipschitz continuity of λ^* , we can also find $\varepsilon > 0$ such that the $(\lambda^* + \varepsilon)$ -flow through $(r, 0)$ crosses $\rho_2 = 1$. By Theorem 16, using any policy, the SC strategy executed on $F \in \mathcal{F}(n, rn)$ must w.h.p. at some stage produce a random formula with $\rho_2 > 1$, and fail to satisfy the input formula.

In short, for any 3-clause density less than ρ_3^* , a random formula can with positive probability be satisfied by the SC strategy equipped with policy σ^* , while for any 3-clause density greater than ρ_3^* , there is no policy σ for which SC satisfies a random formula with positive probability.

6.5 Numerics

Essentially, to compute $\rho_3^* \approx 3.22$, we simply simulated the differential equations given by λ^* : beginning at some point $(0, \rho_3)$, we compute λ^* , take a small step with that slope to arrive at a new point (ρ'_2, ρ'_3) , compute the slope λ^* there, and so forth. The value 3.22 was the largest ρ_3 for which, starting from $(0, \rho_3)$, we observed $\rho_2 < 1$ when $\rho_3 = 2/3$ was crossed. We took care to be conservative: to try to ensure that 3.22 is a bit smaller than the true value of ρ_3^* , and so $\mathcal{F}(n, 3.22n)$ is indeed satisfiable w.h.p. Let us describe a few details.

First, the optimal slopes λ^* depend on solving an knapsack problem of infinite size, with a case for each \bar{A} . However, a suboptimal slope can be computed by limiting the set of cases actively considered to those with probability measure, say, $1 - 10^{-8}$, and in the remaining cases, using a “default policy” $\sigma(\bar{A}) = 1$. This is a manageable computation, and its flows lie above those for λ^* (they are conservative).

Now we consider the issue of the finite step size. Most generally, as one would expect from Lipschitz continuity of the vector field, our computations were stable: from a given initial point, and making several runs with ever-smaller step sizes, the endpoints converged. Moreover, inspection of the simulated flows (see Figure 2) shows them to be concave. If the true flows are indeed concave (which unfortunately seems quite difficult to prove), then a flow lies below its tangent at any point, so taking a finite step size is conservative – the true flow lies below the finite-step simulation.

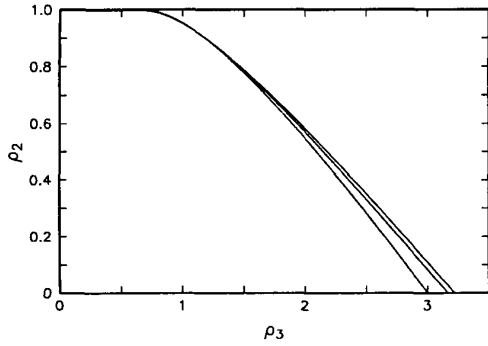


Figure 2. The one-variable algorithm's trajectory in (ρ_2, ρ_3) -space, under three policies. The lowest curve, starting from $\rho_3 = 3.001$, sets $A = 1$ always. The next, starting at 3.166, sets A to minimize the number of unsatisfied 3-clauses, breaking ties by setting $A = 1$. The best, starting at 3.22, is the optimal policy. Near $\rho_2 = 1$, the policies virtually coincide because nearly all moves are forced.

We did not do so, but a rigorously conservative simulation could be obtained by exploiting the Lipschitz continuity: Given the slope λ^* at (ρ_2, ρ_3) we know that the slope is at least $\lambda^* - K\Delta$ for any point within Δ of (ρ_2, ρ_3) . Taking steps of size Δ with slope $\lambda^* - K\Delta$ gives a sequence of points guaranteed to lie above the true flow.

We believe our numerical estimates, of the largest value of ρ_3 for which $(\rho_3, 0)$ is feasible, to be accurate to two digits after the decimal point. In particular, reproducing the result of [11] (that 3.003 is feasible) we show feasibility for 3.001 but fail to show it for 3.002. Table 6.5 shows that when evaluating other policies (and other algorithms, described in the next sections), we reproduce other previously known values to a similar degree of accuracy, and always on the conservative side.

	Algorithm	Our ρ_3	Known ρ_3
1-variable	GUC	3.001	3.003 [11]
	TIE	3.166	3.167 [1]
	Optimal	3.22	
2-variable	TT	3.142	3.145 [2]
	Optimal	3.19	
Mixed 1/2-variable	Optimal	3.26	

7. Other algorithms, other bounds

The procedure we have described is most general, and we have applied it to a number of other algorithms.

Other one-variable algorithms: First, still “picking” only one variable at a time, consider drawing that variable from either a 2-clause or a 3-clause. To optimize this choice,

we augment equations (5)–(7) with a similar set for sampling a 3-clause variable, find the optimal slope λ^* for both sets, and choose the larger one. Along the critical trajectory, we find that it is always better to draw a 2-clause.

At a single (ρ_2, ρ_3) , mixing the strategies can never be advantageous: the choice of strategy can be cast as a nested knapsack problem, and knapsack always has 0/1 optimal solutions. Since choosing a variable at random can be expressed as a mixture of the 2-clause and 3-clause strategies, it is also suboptimal. Thus, our 3.22 bound is the best achievable by *any* myopic, single-variable algorithm.

Two-variable algorithms: However, we can explore algorithms that look at two variables at a time. We were particularly motivated by the recent result of [2], whose algorithm was: select a 2-clause at random, count occurrences of both its literals A and B in 3- and 2-clauses to generate an 8-tuple $(A_2^+, A_2^-, A_3^+, A_3^-, B_2^+, B_2^-, B_3^+, B_3^-)$, and set the pair AB equal to 11, 10, or 01 accordingly. Specifically, [2] used the policy of setting A and B to minimize the number of unsatisfied 3-clauses, thereby showing feasibility of $\rho_3 = 3.145$; this was the best value to date.

We can find an optimal policy for the same strategy. For each 8-tuple \vec{AB} , our “knapsack” selects from three pairs (y_i^j, x_i^j) , corresponding to the three possible settings of A and B . We establish a default policy – set AB to 11 – and for a given \vec{AB} compute the differences between the other policies and the default. The algebraic form of these differences shows that the optimal policy depends only on $A_3^+ - A_3^-$ and so forth, and thus we can gain computational efficiency by reducing the 8-tuple to a 4-tuple. (For the single-variable algorithm, we used the default policy $A = 1$ and reduced each 4-tuple to a pair.)

Specifically, for the default policy $AB = 11$,

$$\mathbb{E}(\Delta n) = -2 \left(1 + \frac{\rho_2}{1 - \rho_2} \right)$$

$$\mathbb{E}(\Delta m_2) = -1 + 2 \left(1 + \frac{\rho_2}{1 - \rho_2} \right) \left(\frac{3}{2} \rho_3 - 2\rho_2 \right)$$

$$\mathbb{E}(\Delta m_3) = -2 \left(1 + \frac{\rho_2}{1 - \rho_2} \right) 3\rho_3.$$

Define \vec{AB} to count appearances *other* than in the chosen clause, and let a_i, b_i denote $A_i^- - A_i^+$ and $B_i^- - B_i^+$ respectively. The differences between $AB = 10$ and the defaults are

$$\mathbb{E}^{10}(\Delta n) - \mathbb{E}^{11}(\Delta n) = b_2 \frac{1}{1 - \rho_2}$$

$$\mathbb{E}^{10}(\Delta m_2) - \mathbb{E}^{11}(\Delta m_2) = -b_2 \frac{\frac{3}{2} \rho_3 - 2\rho_2}{1 - \rho_2} - b_3$$

$$\mathbb{E}^{10}(\Delta m_3) - \mathbb{E}^{11}(\Delta m_3) = b_2 \frac{3\rho_3}{1 - \rho_2},$$

with symmetrical results for $AB = 01$ in lieu of 10.

We find that the optimal two-variable policy is feasible for $\rho_3 = 3.19$, but not much beyond that; that is, the optimal two-variable algorithm is actually less good than the optimal one-variable algorithm. How can this be?

In general terms, by setting two variables at a time, in a single “round” of the algorithm (a free move followed by a series of forced unary clause resolutions) we have more control over the decrease in m_3 and the increase in m_2 (the numbers of 3- and 2-clauses), but we also more rapidly decrease the number of variables n . Since the relevant parameters, ρ_3 and ρ_2 , are the ratios m_3/n and m_2/n , the improvements we make in the numerators may be (and apparently are) more than offset by worsening the denominator.⁴ Also, in a single-variable algorithm, when we reveal two variables (in the course of two free moves), we have four choices for how to set them; each may be set to 1 or to 0. In the two-variable algorithm we have only three choices; the setting 00 would violate the selected clause. Perhaps this explains the result.

Whatever the explanation, we observe (see the table) that another one-variable algorithm, TIE, also dominates its two-variable equivalent, TT. Their policies are to minimize the number of 3-clauses unsatisfied in a free move, breaking ties by setting $A = 1$ (TIE) and $AB = 11$ (TT). (The comparison is a bit unfair since TIE’s $A = 1$ gives a bias towards satisfying 2-clauses, while TT’s $AB = 11$ — symmetric to $AB = 01$ and $AB = 10$ — does not.)

A mixed 1-/2-variable algorithm: A better algorithm is the following. Select a 2-clause, reveal one of its literals, A , and make the usual counts (A_2^+ , A_2^- , A_3^+ , A_3^-). Based on these values, decide whether to set $A = 1$ (policy 1), set $A = 0$ (policy 2), or reveal the companion literal B . In the last case, make the corresponding counts (B_2^+ , B_2^- , B_3^+ , B_3^-), and then decide whether to set AB to 01 (policy 3), 10 (policy 4), or 11 (policy 5).

Note that both the single-variable and the two-variable algorithms are special cases of this mixed algorithm: limiting the choices to 1, 2 gives a single-variable algorithm, while limiting them to 3–5 gives a two-variable algorithm. Thus the optimal policy for the mixed algorithm must do at least as well as that for either of the previous algorithms.

Finding an optimal policy for this algorithm requires a “nested” extension of our knapsack algorithm, in which any (x, y) pair can itself be the result of optimizing another knapsack problem. It is solved by the same method, of using a maximization of a “height” with respect to a given slope λ to find the optimal λ^* . In this case, consider choosing the right policy for a given A -tuple. The values (y_i^1, x_i^1) and (y_i^2, x_i^2) can be written down by analogy with (5)–(7). For the third pair, (y_i^3, x_i^3) , enumerate all B -tuples, and

⁴Actually, in all algorithms that choose variables from 2-clauses, the rate of decrease of 3-clauses is fixedly coupled to that of variables, as $\mathbb{E}(\Delta m_3) = 3\rho_3 \mathbb{E}(\Delta t)$.

(using λ), optimize the “mini-knapsack” with three policy choices (AB equal to 01, 10, or 11) for each B -tuple and the given A -tuple. The corresponding “mini-policy” generates a total contribution (y_i^3, x_i^3) over all B -tuples and the given A -tuple: this is the best we can achieve if we decide to look at B . Now we have a straightforward knapsack problem with three choices in each case, and (still relying on the same λ) we solve it. The resulting policy will have a slope larger than, equal to, or smaller than the “input” λ , and we iterate exactly as described for our regular knapsack problem.

We remark that policy 2 should never be used. If we decide to set $A = 0$, we will in any event be forced to set $B = 1$. If instead of applying policy 2, we reveal \bar{B} and then apply policy 3 regardless, the outcome is the same. But having revealed \bar{B} , the ability to switch between policies 3, 4, and 5 for different \bar{B} ’s will always give some advantage over policy 2.

Again, we establish a default policy, both because the expressions for the difference between two policies are simpler than those for a single policy, and so that some policy (the default) is applied even for values of AB that we fail to consider actively. In this case, we need two levels of defaulting. Globally, we take policy 1 as the default: $A = 1$. For a given A -tuple, when considering policies 3–5, we take $AB = 01$ (policy 3) as the default; even if we enumerated *no* B -tuples, this would be equivalent to policy 2, and any enumeration we do only improves on this. Because of the increased complexity, we enumerate values of AB whose combined probability is only $1 - 10^{-3}$ or more (rather than the $1 - 10^{-8}$ we used in the single-variable case), treating A and B symmetrically.

There is no need to present the equations for this algorithm explicitly. The expectations for the $A = 1$ default policy are as in the single-variable algorithm; the expectation for an $AB = 01$ default mini-policy is just the single-variable difference between $A = 0$ and $A = 1$; and the differences between various AB policies and the $AB = 01$ default are just as for the two-variable algorithm. Scavenging from these old cases the computer code, as well as the equations, minimized the introduction of new bugs.

The resulting algorithm shows 3.26 to be feasible.

8. Conclusions

Previous works considered “static” algorithms in order to establish lower bounds for r_3 , exploiting something similar to the predictability of trajectories in (ρ_2, ρ_3) -space. Working in (ρ_2, ρ_3) -space is a conceptual simplification introduced here, allowing us to consider algorithms that optimize their policies continually. In this setting we proved that optimal policies correspond to “flattest” trajectories, and we reduced computing such policies to solving instances of a “max-density multiple-choice knapsack” prob-

lem. We showed how such knapsack problems can be solved efficiently and, further, how the structure of optimal knapsack solutions characterizes optimal policy choices in terms of “expected utility”.

By producing optimal myopic algorithms, we also learn where and where not to look for further improvement. No myopic strategy that looks at a single card, or a single 2-clause, can give any improvement. There is also no advantage in selecting multiple literals randomly and independently, as they have probability $o(1)$ of interacting. However, where we revealed a clause’s literal A and decided whether to reveal the other literal in the same clause, we could equally well consider revealing any or all of A ’s companion literals in *all* 2-clauses (and/or 3-clauses). Our optimization method applies here (if the number of cards revealed is capped at a constant), but implementing the optimization appears daunting. We have no idea whether the improvement would be small or large; it is an interesting question.

However, we feel that the real weakness of the myopic approach is the very limited availability of variable-degree information. Card-pointing in free moves gets at such information indirectly: by guaranteeing that the selected variable has degree at least 1, it weakly biases the selection towards variables of high degree. Considerably stronger bounds might be obtained from analyzing an algorithm which capitalizes on degree information; this would entail working with random formulas conditioned upon a degree sequence.

On another front, using the non-rigorous “replica” method of statistical mechanics, Monasson and Zecchina [19] predicted that for any $\epsilon > 0$, the point $(1 - \epsilon, 0.71)$ in (ρ_2, ρ_3) -space is feasible. If so, this gives a corresponding improvement in our bounds ρ_3^* for r_3 : ρ_3^* becomes the ρ_3 -intercept of the flow through $(1, 0.71)$ rather than $(1, 2/3)$. However, it can be shown that every myopic algorithm fails w.h.p. if started at $(1 - \epsilon, \rho_3)$, for $\rho_3 > 2/3$ and ϵ sufficiently small. Thus, resolving whether $2/3$ is tight — an interesting question in its own right — requires methods outside the realm of myopic algorithms.

Acknowledgments

We are grateful to Don Coppersmith and Alan Frieze for many helpful discussions. Among other things, Don discovered the knapsack’s λ , and Alan suggested perturbing it by ϵ . We also thank the anonymous reviewers.

References

- [1] D. Achlioptas. A survey of lower bounds for random 3-SAT via differential equations. *Theoret. Comput. Sci.*, to appear. Available as www.research.microsoft.com/~optas/lbsurvey.ps.
- [2] D. Achlioptas. Setting two variables at a time yields a new lower bound for random 3-SAT. In *32nd Annual ACM Symposium on Theory of Computing (Portland, OR, 2000)*, pages 28–37. ACM, New York, 2000.
- [3] D. Achlioptas, L. M. Kirousis, E. Kranakis, and D. Krizanc. Rigorous results for random $(2+p)$ -SAT. *Theoret. Comput. Sci.*, to appear.
- [4] B. Bollobás, C. Borgs, J. Chayes, J. H. Kim, and D. B. Wilson. The scaling window of the 2-SAT transition. Manuscript, 1999.
- [5] A. Z. Broder, A. M. Frieze, and E. Upfal. On the satisfiability and maximum satisfiability of random 3-CNF formulas. In *4th Annual ACM-SIAM Symp. on Disc. Alg. (Austin, TX, 1993)*, pages 322–330. ACM, New York, 1993.
- [6] M.-T. Chao and J. Franco. Probabilistic analysis of two heuristics for the 3-satisfiability problem. *SIAM J. Comput.*, 15(4):1106–1118, 1986.
- [7] V. Chvátal and B. Reed. Mick gets some (the odds are on his side). In *33th Annual Symposium on Foundations of Computer Science (Pittsburgh, PA, 1992)*, pages 620–627. IEEE Comput. Soc. Press, Los Alamitos, CA, 1992.
- [8] O. Dubois, Y. Boufkhad, and J. Mandler. Typical random 3-SAT formulae and the satisfiability threshold. In *11th Annual ACM-SIAM Symp. on Disc. Alg. (San Francisco, CA, 2000)*, pages 126–127. ACM, New York, 2000.
- [9] W. Fernandez de la Vega. On random 2-SAT. Manuscript, 1992.
- [10] E. Friedgut. Necessary and sufficient conditions for sharp thresholds of graph properties, and the k -SAT problem. *J. Amer. Math. Soc.*, 12:1017–1054, 1999.
- [11] A. M. Frieze and S. Suen. Analysis of two simple heuristics on a random instance of k -SAT. *J. Algorithms*, 20(2):312–355, 1996.
- [12] M. R. Garey and D. S. Johnson. *Computers and Intractability*. Freeman, San Francisco, CA, 1979.
- [13] A. Goerdt. A threshold for unsatisfiability. *J. Comput. System Sci.*, 53(3):469–486, 1996.
- [14] R. L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, 1:132–133, 1972.
- [15] S. Janson, Y. C. Stamatiou, and M. Vamvakari. Bounding the unsatisfiability threshold of random 3-SAT. *Random Structures Algorithms*, 17(2):103–116, 2000.
- [16] T. G. Kurtz. Solutions of ordinary differential equations as limits of pure jump Markov processes. *J. Appl. Probability*, 7:49–58, 1970.
- [17] R. Monasson and R. Zecchina. Entropy of the K -satisfiability problem. *Phys. Rev. Lett.*, 76(21):3881–3885, 1996.
- [18] R. Monasson and R. Zecchina. Statistical mechanics of the random K -satisfiability model. *Phys. Rev. E* (3), 56(2):1357–1370, 1997.
- [19] R. Monasson and R. Zecchina. Tricritical points in random combinatorics: the $(2+p)$ -SAT case. *J. Phys. A: Math. and Gen.*, 31(46):9209–9217, 1998.
- [20] J. Palis, Jr. and W. de Melo. *Geometric Theory of Dynamical Systems*. Springer-Verlag, New York, 1980.
- [21] L. Trevisan, G. B. Sorkin, M. Sudan, and D. P. Williamson. Gadgets, approximation, and linear programming. *SIAM J. Comput.*, 29(6):2074–2097, Dec. 2000.
- [22] N. C. Wormald. Differential equations for random processes and random graphs. *Ann. Appl. Probab.*, 5(4):1217–1235, 1995.