Backdoor Set Detection for 3CNF Formulas

Andrew Kaploun,
Supervised By Serge Gaspers

School of Computer Science
University of New South Wales, Sydney

Thesis A, Trimester 3 2019

Table of Contents

Table of Contents

Parameterized Algorithms

Algorithms

- Input of size $n \in \mathbb{N}$
- Runtime described by $f(n)$, for some function $f : \mathbb{N} \to \mathbb{N}$

Parameterized Algorithms

- Input of size $n \in \mathbb{N}$
- Parameter $k \in \mathbb{N}$
- Can describe the runtime by $f(k, n)$, for some function $f : (\mathbb{N}, \mathbb{N}) \to \mathbb{N}$

Definition

"O^* notation ignores polynomial factors in the input size" (COMP6741)
$O^*(f(k)) \equiv O(poly(n) \cdot f(k))$

Vertex Cover

The size of the solution is often a parameter, for which Vertex Cover is a classic example:

Input

A graph $G = (V, E)$ and natural number k

Parameter

k

Question

Is there a set of not more than k vertices such that $\forall e \in E, \exists v \in G$ such that v is adjacent to e.

Example: Branching Algorithm

```
Vertex Cover((V, E), k):

    if |E| = 0:
        return True

    if k = 0:
        return false

    Pick an arbitrary edge uv in E

    return
        Best(Vertex Cover(G-u, k-1),Vertex Cover(G-v, k-1))
```

Example: Branching Algorithm

Analysis

We bound the number of leaves in the search tree. Since each tree node takes polynomial time to process, the runtime is O^*(number of leaves in the search tree). Let $T(k)$ be the number of leaves in the search tree if we have budget k.

$$T(k) \leq T(k-1) + T(k-1).$$

Hence since we branch into two at each point in the search tree, and there are at most k layers, we can say that there are at most $O(2^k)$ nodes, thus we have a run time bound by $O^*(2^k)$.

SAT

Literal

A literal is a boolean variable with or without a negation. (eg. $x, \neg y$)

SAT

- Input: A logical formula ϕ consisting of conjunctions ($\wedge$), disjunctions ($\vee$), and literals.
- Question: Is there an assignment of true and false values such that ϕ is true?

Disjunctive Clause

Input: A disjunctive clause, or simply a clause, is a set of literals connected by a disjunction ($\vee$).

SAT

Conjunctive Normal Form

A formula ϕ in CNF or Conjunctive Normal Form consists of clauses separated by conjunctions.

Example

$(x \vee a \vee b) \wedge (\neg x \vee c \vee d) \wedge (\neg x \vee e \vee f)$

Assignment

If we have a formula ϕ and a set of assignments τ, we denote ϕ with the assignments in τ substituted in by $\phi[\tau]$. (And clean up stray true clauses)

Example

$(x \vee a \vee b) \wedge (\neg x \vee c \vee d) \wedge (\neg x \vee e \vee f)[x \leftarrow \mathit{false}] = (a \vee b)$

SAT Classes

SAT Class

A class of SAT is a set of SAT formulae that satisfy some property.

SAT Class Examples

A class of SAT is a set of SAT formulae that satisfy some property.

- 3-CNF: Formulae in CNF with clauses containing at most 3 variables.
- 2-CNF: Take a guess!
- 0-Val: Each clause has at least one negative literal (one variable of the form $\neg x$.) Note that if a formula $\phi \in 0 - \mathit{val}$, an entirely negative assignment satisfies ϕ.
- The Null Formula: True

SAT Backdoors

Weak Backdoors

A weak backdoor from class $\mathcal{C}_1$ to $\mathcal{C}_2$ for a formula ϕ is a truth assignment $\tau : Var(X) \rightarrow \{0, 1\}$ such that $\phi[\tau] \in \mathcal{C}_2$, and $\phi[\tau]$ is satisfiable.

Backdoors

We also call weak backdoors ***backdoors*** or simply *WB*.

Backdoor Algorithms

Input

A formula $\phi \in \mathcal{C}_1$ and natural number k.

Parameter

k

Question

Does there exist a backdoor from ϕ to a formula in $\mathcal{C}_2$ that consists of an assignment of no more than k variables?

Example

$WB(3CNF, NULL)$ with parameter k asks if we can make an assignment of no more than k variables that satisfies some satisfiable formula input into the algorithm.
For our purposes, assume no clause contains a variable both negatively and positively (eg. $(x \vee \neg x)$)

d-Hitting-Set

Input

A collection C of subsets of a finite set S, where $|C| \leq d$, and an integer k

Parameter

k

Question

Is there a subset $S' \subseteq S$ with $|S'| \leq k$ that requires S' to contain at least one element from each subset in C?

Example

$\{(a, b, c), (c, d, e)\}, k = 1$

Table of Contents

Summary

Parameterized Measure and Conquer

Evolutions of Parameterized Measure and Conquer have often manifested in improvements in the runtime d-Hitting-Set.

Local Search

Local Search has been applied to many problems, with *SAT* as a significant example.

Formalisation of Parameterized Measure and Conquer

Properties

We can use certain properties to branch less and prove the tree is smaller. For example, if there is a 2-set in 3-hitting-set, we can branch into 2 vertices.

Parameterized Measure and Conquer

Niedermeier and Rossmanith [1999] gave an early form of parameterized measure and conquer that gave an $O^*(2.27^k)$ runtime. They encoded the state in the equations

$$T(k) = 1 + T(k-1) + T(k-2) + B(k-1)$$

$$B(k) = 1 + B(k-1) + T(k-1)$$

The 'state', i.e. whether it has a 2-set, is encoded in whether the equation is $B(k)$ or $T(k)$.

Formalisation of Parameterized Measure and Conquer

More detailed states in Fernau ($O^*(2.1788^k)$)

Fernau [2004] did a more detailed case analysis where for $d-$Hitting-Set, if there are at least n 2-sets,

$$T_d^n(k) := \text{The number of leaves in the search tree with a budget of } k$$

Note

These methods still only have the capability to take into account one type of 'measure'.

Wahlström's Method ($O^*(2.0755^k)$)

Walhström

Walhström [2007, PhD Thesis] refined an approach for exact exponential algorithms by Eppstein [2004]. He gave an approach for assigning many weights, for both parameterized and exact exponential problems. He defines states of a problem F as

$$S(F) = k \iff F \text{ is in state } S_k.$$

Then we can define a measure to take into account the parameter and the state for the weight

$$f(F) = n(F) - \psi(S(F)).$$

k-Leaf-Spanning-Tree

Input

A graph G, a natural number k.

Parameter

k

Question

Does G contain a subgraph that is a spanning tree with k leaves?

Similar Technique

k-Leaf-Spanning-Tree

This problem was extensively studied since 1988 when it was proven to be FPT. ($O^*(17k^4!)$ [1989]).
Kneis et al. [2008] proved an $O^*(4^k)$ bound, and Daligault et al. [2008] improved this to $O^*(3.72^k)$.
Fernau, Kneis et al. [2010] used the same idea as above for an exact exponential algorithm, with a measure that used the sizes of

- leaf nodes
- internal nodes
- branching nodes
- floating vertices: vertices that are leaves, but not yet 'attached' to the tree
- free vertices

.

Local Search for k-sat

Randomized SAT Algorithm

Schöning [1999] gave a very simple algorithm for randomized SAT.

Randomized SAT Algorithm

```
SAT(A formula phi over n variables):
    Randomly pick an assignment for phi
    While phi is unsatisfied, repeat 3n times:
        Pick an unsatisfied clause C uniformly at random
        Pick a literal x from C uniformly at random
        'flip' x's underlying variable to be true
```

Local Search for k-sat

Randomized SAT Algorithm

Schöning [1999] proved that if we repeat this algorithm, the expected value of the runtime is

$$O^*\left(\left(\frac{2(k-1)}{k}\right)^n\right),$$

and thus $O^*(1.334^n)$ for $k = 3$.

Derandomization General Outline

Hamming Distance

The hamming distance $\mathcal{H}$ between two equal length bitstrings is the number of positions in which they differ.

Hamming Ball

$B_{\mathcal{H}}(s, n)$ Denotes the set of all bitstrings no more than n hamming distance from string s.

General Idea

- View an assignment of true and false values as a bitstring of ones and zeroes.
- Prove that if we start at a random assignment and there exists a satisfying assignment within a Hamming Ball of some size, that we will find it within some fixed number of steps of the randomized procedure.
- Thus, we can bound the number of times we call our procedure.

Derandomization Results

First Derandomization

Dantsin, Goerdt, Hirsch, Kannan, Kleinberg, Papadimitriou, Raghavan and Schöning [2002] gave a derandomization that gives a deterministic runtime of

$$O^*\left(\left(\frac{2k}{k+1}\right)^n\right),$$

which for $k = 3$ is $O^*(1.5^n)$.

Fastest Derandomization

Moser & Scheder [2011] Derandomized this algorithm to prove a bound of

$$O^*\left(\left(\frac{2(k-1)}{k}\right)^{n+o(n)}\right),$$

and thus $O^*(1.334^{n+o(n)})$ for $k = 3$.

WB(3CNF, 0-Val)

Raman, Shankar [2013] used a non-measure and conquer branching analysis to improve on the trivial $O^*(3^k)$ trivial bound, giving an algorithm that runs in $O^*(2.85^k)$.

Recommendation

They recommended in the conclusion that perhaps it may be a potential research problem to find a parameterized bound for $WB(3CNF, Null)$.

Note

It can be easily intuitively observed that $WB(3CNF, 2CNF)$ can be reduced to $3 - Hitting - Set$, which is what Misra, Ordyniak, Raman, and Szeider [2013] proved in a summary of upper and lower bounds on backdoors.

This lets us observe the relationship between $3 - Hitting - Set$ and $WB(3NCF, NULL)$, by seeing that $WB(3CNF, 2CNF)$ is a special case of $WB(3NCF, NULL)$ where all variables only occur positively.

Table of Contents

Definitions

Definition 1.1

A literal x in a clause is referred to as an (a, b) literal if x occurs a times in ϕ, and $\neg x$ occurs b times in ϕ.

Definition 1.2

(a, b) variables are made up of (a, b) and (b, a) literals.

Example

$(x \vee a \vee b) \wedge (\neg x \vee c \vee d) \wedge (\neg x \vee e \vee f)$

Definitions

Definition 1.4

We say that a literal of the form (a', b') is of the form

- $(a+, b')$ if $a \leq a'$
- $(a', b+)$ if $b \leq b'$
- $(a+, b+)$ if both of the above conditions hold

Note

In our algorithm, τ' will be a set containing literals that we guarantee we will not set to true.

Definition 1.4

A semi-2-clause is a 3-clause where 1 literal is in τ'.

Lemmas

Lemma 1

If every variable is of the form $(1+, 0)$, then we can solve $WB(3CNF, NULL)$ in $O^*(2.0755^k)$.

Proof.

Reduction to 3-Hitting-Set. □

Lemmas

Lemma 2

If every variable is of the form $(1, 1)$, then we can solve $WB(3CNF, NULL)$ in polynomial time.

Proof.

Proof. First, show that there exists a satisfying assignment of size $|\mathcal{C}|$, where $\mathcal{C}$ is the set of clauses of ϕ, if and only if the formula is satisfiable.
For one side of the inequality, note $|\tau| \geq |\mathcal{C}|$ since each variable assignment can only satisfy one clause.
Then, to obtain an assignment τ such that $|\tau| \leq |\mathcal{C}|$, take any formula τ which has size greater than $|\mathcal{C}|$. While $|\tau| \leq |\mathcal{C}|$, pick an arbitrary variable from a clause that has more than 1 satisfied literal, and remove it from τ.
Tovey [1984] proved there exists such an assignment, and that we can find in polynomial time. □

Trivial Reduction Rules

Intuition

The reduction rules will reduce the problem to something where we can branch better than the trivial 3-direction branching. i.e. $((1, 2+) \vee (1+, 1+) \vee (1+, 1+))$ or $((2+, 2+) \vee y \vee z)$

Rule 1

If there exists a clause with only one literal, add the variable to τ so as to make the literal true.

Rule 2

If the same literal occurs more than once in any clause, remove the duplicate occurrences. (eg. $(\neg x \vee \neg x \vee y) \rightarrow (\neg x \vee y)$)

Rule 3

If every variable is of the form $(1, 1)$, apply lemma 2.

Non-Trivial Reduction Rules

Rule 4

If ϕ has only variables of the form $(1, 1)$ and $(1+, 0)$, and we have a clause that contains a $(1, 1)$ literal and a $(1+, 0)$ literal, delete the $(1, 1)$ literal.

Rule 5

If ϕ has only variables of the form $(1, 1)$ and $(1+, 0)$, and no clauses have literals of both forms:

- If we have l clauses of $(1, 1)$ variables, by Lemma 2 we can call our algorithm with parameters

$$G \leftarrow \phi - \{\text{clauses with } k \text{ variables}\}, k \leftarrow k - l$$

Rule 6

If a clause contains a $(1, 2+)$ literal and a $(1+, 0)$ literal, assign the $(1+, 0)$ literal true in τ.

Branching Rules & Analysis

Definition

Let $T_n(k)$ denote the runtime of the algorithm for an instance where

- The parameter is k.
- (# of 2-clauses) + (# of semi-2-clauses) $\geq n$

Rule 1

If there is a 2-clause with literals x and y, branch on

- Adding a truth assignment that makes x true to τ
- Adding a truth assignment that makes y true to τ

Analysis for Rule 1

$$T_n(k) \leq T_{n-1}(k-1) + T_{n-1}(k-1)$$

Branching Rules & Analysis

Example

$$((2+, 2+) \vee y \vee z)$$

Rule 2

If ϕ contains a clause that contains a literal x of form $(2+, 2+)$ branch on the following:

- Add an assignment to τ that makes x true.
- Add x to τ'

Analysis for Rule 2

$T_n(k) \leq T_{n+2}(k) + T_{n+2}(k-1)$

Branching Rules & Analysis

Example

$$((1, 2+) \vee (1+, 1+) \vee (1+, 1+))$$

Rule 3

Note that after exhaustively applying the reduction rules, if we have a $(1, 2+)$ literal x, x shares a clause with only literals of the form $(1+, 1+)$. Thus, denote our clause by $(x \vee y \vee z)$ branch on

- Add an assignment to τ that makes x true.
- Add an assignment to τ that makes y true.
- Add an assignment to τ that makes z true.

Analysis for Rule 3

$T_n(k) \leq T_{n+2}(k-1) + 2T_{n+1}(k-1)$

Algorithm

```
WB{3CNF, NULL}(phi, k, tau'):

    Apply the reduction rules exhaustively

    if Branching Rule 1 applies:
        Apply Branching Rule 1
    else if Branching Rule 2 applies:
        Apply Branching Rule 2
    else if Branching Rule 3 applies:
        Apply Branching Rule 3
```

Branching Rules & Analysis

Theorem

$$T_n(k) \leq max \begin{cases} T_{n+2}(k) + T_{n+2}(k-1) \\ T_{n+2}(k-1) + 2T_{n+1}(k-1) \end{cases}$$

Applying Rule 1,

$$T_n(k) \leq max \begin{cases} 4T_n(k-2) + 4T_n(k-3) \\ 4T_n(k-3) + 4T_n(k-2) \end{cases}$$

Thus a suitable function is an exponential function with base c such that

$$c^k \leq 4c^{k-2} + 4c^{k-3} \iff c^3 \leq 4c + 4.$$

So we can pick $c = 2.38298$ giving us the bound

$$O^*(2.38298^k).$$

Table of Contents

Explore Earameterized Measure and Conquer

WB(3CNF, NULL)

We obviously have an advantage when we have 2-clauses and semi-2-clauses. We can attempt to find other areas that give us an advantage.
Allows us to avoid a big case analysis.

WB(3CNF, 0-Val)

One possible parameter we can explore is having a set of 'unassigned' variables, a set of 'definitely not true', and a set of 'definitely not false' variables to aid in the analysis. This has worked well for problems where you have to 'pick' some number of variables, like k-leaf-spanning tree.

Local Search

WB(3CNF, 0-Val)

Similar to the local search for SAT, we can start with an all 0 assignment and randomly satisfy unsatisfied clauses with a 1. Then we can apply a strategy similar to the randomized SAT algorithm.