Санкт-Петербургский государственный университет

Системное программирование

Группа 22.М07-мм

Разработка MaxSMT-решателя, поддерживающего режим портфолио

Фомина Виктория Викторовна

Отчёт по производственной практике
в форме «Решение»

Научный руководитель:
старший преподаватель кафедры системного программирования, к.т.н., Ю. В. Литвинов

Консультант:
преподаватель ИПКН ИТМО, В. Соболь

Санкт-Петербург
2024

Оглавление

Введение

В таких сферах анализа программного обеспечения, как верификация, символьное исполнение, генерация тестов, широко применяется понятие выполнимости формул [7, 4, 10]. В частности, нередко возникает необходимость решения задачи SAT (Boolean satisfiability problem) [18], которая заключается в определении выполнимости логических формул или, другими словами, в определении, можно ли присвоить значения переменным логического выражения таким образом, чтобы получить истину.

Однако в реальной практике часто приходится работать не только с логическими формулами, но и с формулами, составленными из сложных объектов. Строить такие формулы позволяют теории. Они обобщают логические формулы на формулы из сложных объектов. В качестве сложных объектов могут выступать целые и вещественные числа, а также различные структуры данных, такие как массивы, битовые вектора, строки. Задача выполнимости формул в теориях называется SMT (Satisfiability modulo theories) [14]. Для её решения используются специализированные программные инструменты, которые называются SMT-решателями (SAT-решатели для задачи SAT).

Представим, что наша цель — сгенерировать юнит-тесты. В этом про Задача SMT помогает генерировать регресионные тесты, покрывающие ветви исполнения. При этом сгенерированные тесты могут быть ”некрасивыми”: создавать массивы огромного размера и заполнять их числами с большими абсолютными значениями. Можно добавить дополнительные ограничения на размеры генерируемых массивов и значения их элементов, при этом наделяя эти ограничения приоритетами. Например, ограничению, согласно которому размер массива не должен превышать 10000, можно присвоить более высокий приоритет, чем ограничению, требующему, чтобы все элементы массива были положительными. Останется решить задачу оптимизации с ограничениями, порождаемыми прохождением по ветви исполнения и дополнительными ограничениями для более ”красивых” тестов.

Формальным представлением задачи оптимизации является $MaxSMT$ — оптимизационная версия задачи выполнимости формул в теориях ($MaxSAT$ — задачи SAT) [17, 11, 12, 13, 5].

Задачу $MaxSMT$ можно поставить так.

- Имеется набор утверждений H, выполнимость (здесь и далее имеется в виду выполнимость конъюнкции утверждений) которых равносильна наличию решения задачи (как в задаче SMT).
- Имеется набор утверждений S, которые могут не выполняться. При этом каждому утверждению s из этого набора соответствует приоритет его выполнения — вес $weight(s)$.
- Требуется найти решение — набор утверждений K из всех выполнимых утверждений из S: значение $\sum_{s \in K} weight(s)$ является наибольшим возможным.

В ходе решения задачи MaxSMT необходимо многократно решать задачу SMT. Стоит отметить, что сама задача SMT представляет собой NP-трудную задачу для нетривиальных теорий и, в общем случае, является неразрешимой.

Разные SMT-решатели поддерживают различные наборы теорий. Чтобы MaxSMT-решатель мог работать с широким спектром теорий, важно иметь возможность взаимодействовать с различными SMT-решателями. В этом контексте требуется промежуточный слой, который обеспечивает единообразное взаимодействие с разными SMT-решателями. Существует Kotlin-библиотека KSMT, которая предоставляет унифицированный API для работы с большим количеством SMT-решателей.

Ввиду высокой вычислительной сложности и неразрешимости в общем случае задачи выполнимости формул в теориях, SMT-решатели обычно используются с ограничением по времени. По этой причине имеет смысл использовать с ограничением по времени и MaxSMT-решатель. В условиях ограниченного времени целесообразно искать не

оптимальное, а субоптимальное решение — лучшее из найденных за отведенный промежуток времени.

В то же время SMT-решатели показывают разную производительность даже на идентичных наборах входных данных[1]. Это подчеркивает важность режима портфолио для MaxSMT-решателя. Режима, позволяющего одновременного запускать несколько MaxSMT-решателей, работающих с разными SMT-решателями, и возвращать результат от того решателя, который завершил работу быстрее.

Написанию решателя задачи $MaxSMT$ с возможностью поиска субоптимальных решений, поддерживающего режим портфолио, и посвящена данная работа. Реализация $MaxSMT$-решателя планируется в рамках работы над Kotlin-библиотекой KSMT[2], предоставляющей унифицированный API для взаимодействия с набором SMT-решателей.

[1]SMT-COMP 2023 — https://smt-comp.github.io/2023/ (дата обращения: 04.05.2024)

[2]KSMT — https://ksmt.io/ (дата обращения: 24.12.2023)

Постановка задачи

Целью работы является разработка MaxSMT-решателя с поддержкой режима портфолио в рамках работы над Kotlin-библиотекой KSMT.

Для ее выполнения были поставлены следующие задачи.

1. Исследовать существующие алгоритмы, решающие задачу MaxSAT с возможностью адаптации под решение MaxSMT с поиском субоптимальных решений.
2. Выбрать и адаптировать алгоритм под решение задачи MaxSMT.
3. Реализовать выбранный алгоритм в библиотеке KSMT, поддержав режим портфолио.
4. Оценить корректность реализации алгоритма на основе тестового набора данных.
5. Сравнить скорость работы MaxSMT-решателя в режиме портфолио и без.

1 Обзор

Введем набор определений и обозначений, необходимых для дальнейшего понимания работы. Затем перейдем к рассмотрению подходов к решению задачи $MaxSMT$ и обзору существующих $MaxSMT$-решателей.

1.1 Определения и обозначения, используемые в работе

Определим термины, специфические для рассматриваемой предметной области, а также введем обозначения, которые будут использоваться в работе. Отметим, что некоторые термины не будут определены в текущем разделе, а будут введены в момент первого использования в работе.

Для того, чтобы формально поставить задачу $MaxSMT$, необходимо определить такие понятия как язык и формула языка первого порядка, поставить задачу SMT, а также определить понятие модели.

Определение 1. *Введем определение сигнатуры языка первого порядка.* ***Сигнатура*** *— это тройка* $\Sigma = (\Sigma_f, \Sigma_p, ar)$*:*

- Σ_f *— множество функциональных символов;*
- Σ_p *— множество предикатных символов;*
- ar *— функция:* $\Sigma_f \cup \Sigma_p \to \mathbb{N}$ *(арность).*

Определение 2. *Зафиксируем сигнатуру* Σ *и счетное множество (предметных) переменных* ν*:* $\nu \cap \Sigma_f = \nu \cap \Sigma_p = \emptyset$*.* ***Множество термов*** $T(\Sigma, \nu)$ *— наименьшее множество для которого верно:*

- $\nu \subseteq T(\Sigma, \nu)$*;*
- $f \in \Sigma_f,\ ar(f) = n\ \ и\ t_1, ..., t_n \in T(\Sigma, \nu) \Rightarrow f(t_1, ..., t_n) \in T(\Sigma, \nu)$*.*

Определение 3. *Определим, что является* ***атомарной формулой****.*

- $p \in \Sigma_p,\ ar(p) = n\ и\ t_1, ..., t_n \in T(\Sigma, \nu) \Rightarrow p(t_1, ..., t_n)$ — *атомарная формула.*

- $True\ и\ False$ — *атомарные формулы.*

Определение 4. ***Язык первого порядка*** *(далее язык)* $FOL(\Sigma, \nu)$ — *наименьшее множество, для которого верно:*

- *любая атомарная формула* $\in FOL(\Sigma, \nu)$*;*

- $\varphi \in FOL(\Sigma, \nu) \Rightarrow \neg\varphi \in FOL(\Sigma, \nu)$*;*

- $\varphi_1, \varphi_2 \in FOL(\Sigma, \nu) \Rightarrow \varphi_1 \wedge \varphi_2 \in FOL(\Sigma, \nu)$

- $\varphi_1, \varphi_2 \in FOL(\Sigma, \nu) \Rightarrow \varphi_1 \vee \varphi_2 \in FOL(\Sigma, \nu)$

- $\varphi \in FOL(\Sigma, \nu),\ x \in \nu \Rightarrow \forall x : \varphi \in FOL(\Sigma, \nu)$

- $\varphi \in FOL(\Sigma, \nu),\ x \in \nu \Rightarrow \exists x : \varphi \in FOL(\Sigma, \nu)$

Определение 5. ***Формулы языка первого порядка*** *(далее формулы)* — *элементы* $FOL(\Sigma, \nu)$*.*

Теперь поговорим о семантике языка первого порядка: о том как интерпретируются формулы.

Определение 6. *Введем понятие структуры.* ***Структурой*** M *называется:*

- *непустое множество* $|M|$*, называемое носителем;*

- *интерпретация функциональных символов* $M(f) : |M|^{ar(f)} \to M$*;*

- *интерпретация предикатных символов* $M(p) \subseteq |M|^{ar(p)}$*.*

Определение 7. *Определим понятие оценки.* ***Оценкой*** *называется тотальное отображение* $v : \nu \to |M|$*.*

Определение 8. *Формула* φ *истинна в структуре* M*, если для всех оценок* v*,* $M \vDash_v \varphi$*. В таком случае* M *называется* ***моделью*** φ *(обозначается* $M \vDash \varphi$*).*

Определение 9. *Множество* $\Gamma \subseteq FOL(\Sigma, \nu)$ ***семантически замкнуто****, если* $\forall \varphi \in FOL(\Sigma, \nu) : \Gamma \models \varphi$*, верно* $\varphi \in \Gamma$*.*

Определение 10. ***Теория первого порядка*** *(далее теория) в языке* $FOL(\Sigma, \nu)$ *— любое семантически замкнутое множество предложений в* $FOL(\Sigma, \nu)$*.*

Определение 11. ***Формула*** φ ***выполнима в теории*** $\mathcal{T}$*, если* $\exists M$ *— модель теории* $\mathcal{T}$*, в которой* φ *истинна (обозначается* $M \models_{\mathcal{T}} \varphi$*).*

Во введение задача $MaxSMT$ поставлена достаточно абстрактно и не достаточно формально. Это сделано для того, чтобы сторонний читатель мог понять о чем повествует работа, не вникая в большое количество специализированной теории. Теперь, когда мы определили все необходимые понятия для формальной постановки задачи, сделаем это — поставим задачу $MaxSMT$ формально.

Определение 12. *Задачу* $MaxSMT$ *поставим так.*

Даны наборы утверждений H *и* S *в теории* $\mathcal{T}$*, оснащенные весами (вес утверждения* k *будем обозначать* $weight(k)$*):*
$\forall h \in H \ weight(h) = \infty \ \wedge \ \forall s \in S \ \exists w \in \mathbb{N} : weight(s) = w.$

$\forall h \in H \ \bigwedge\limits_{h \in H} h$ *выполнима равносильно наличию решения задачи (интерпретация веса* ∞*).*

Пусть $K \subseteq S$ *— набор выполнимых утверждений из* S*: значение* $\sum\limits_{s \in K} weight(s)$ *наибольшее возможное.*

Требуется найти модель M *формулы* $\bigwedge\limits_{h \in H} h \wedge \bigwedge\limits_{k \in K} k.$

И хотя в постановке задачи $MaxSMT$ мы полагаем, что утверждениям из S сопоставляются натуральные веса — это не обязательное требование. Однако такая постановка задачи удобна в рамках рассматриваемой работы.

Обозначение 1. *Будем обозначать буквой* H *множество утверждений в теории* $\mathcal{T}$*, выполнимость которых равносильна наличию решения задачи* $MaxSMT$ *(утверждения с весом* ∞*).*

Обозначение 2. *Будем обозначать буквой S множество утверждений в теории $\mathcal{T}$.*

Определение 13. *Дан набор утверждений K в теории $\mathcal{T}$: формула $\bigwedge\limits_{k \in K} k$ невыполнима. Множество $core \subseteq K$: формула $\bigwedge\limits_{r \in core} r$ невыполнима называется* ***ядром*** *(от англ. unsatisfiable core).*

Определение 14. *Ядро $core$ называется* ***минимальным****, если $\forall R \subset core \bigwedge\limits_{r \in R} r$ выполнима.*

1.2 Подходы к решению задачи $MaxSMT$

Существует три подхода к решению задачи $MaxSMT$.

- Основанный на итеративном улучшении моделей.
  - Ключевая идея подхода состоит в итеративном улучшении модели — кандидата на оптимальное решение.
- Основанный на использовании ядер.
  - Основная идея подхода заключается в использовании ядер в роли сущностей, направляющих к решению.
- Основанный на комбинации использования ядер и моделей.
  - Ядра все еще служат в качестве сущностей, направляющих к решению, при этом используются модели для получения промежуточных решений.

1.3 Существующие $MaxSMT$-решатели

Посмотрим какие $MaxSMT$-решатели представлены в сообществе на сегодняшний день.

Насколько известно, к настоящему времени не представлено ни одного $MaxSMT$-решателя, умеющего переключаться между SMT-решателями в процессе решения задачи. В современных решателях задача $MaxSMT$ реализуется в рамках конкретного SMT-решателя.

Наиболее популярным $MaxSMT$-решателем является Z3 — решатель с открытым исходным кодом, разрабатываемый в компании Microsoft. Этот решатель поддерживает субоптимальность. Важным преимуществом Z3 является умение работать с большим набором теорий, включая такие популярные теории как битовые вектора, массивы, линейная целочисленная и рациональная арифметики, неинтерпретированные функции, числа с плавающей точкой.

Существует также решатель OptiMathSAT [16], основанный на решателе с закрытым исходным кодом MathSAT5 [19]. OptiMathSAT позиционируется решателем задачи OMT (англ. optimization modulo theories), являющейся расширением SMT. Задача OMT ищет решение, оптимальное по отношению к некоторым целевым функциям. Задача $MaxSMT$, целевую функцию которой легко выделить, несложно сводится к OMT [16]. Тем не менее решение задачи $MaxSMT$ через сведению ее к OMT — неэффективный подход, что подтверждают авторы OptiMathSAT в статье [15]. Отметим, что OptiMathSAT поддерживает субоптимальность.

1.3.1 Подход к решению задачи $MaxSMT$, основанный на получении моделей

Одним из классических алгоритмов решения задачи $MaxSMT$, основанных на итеративном улучшении модели, является Алгоритм 1 WMax [3], реализованный в $MaxSMT$-решателе νZ [2] — части Z3 [6].

На вход алгоритму подаются множества H, S, W: $\forall s \in S\ \exists w \in W : weight(s) = w$. Сначала строится формула F, процесс преобразования которой приведет к оптимальному решению (строки 1-2): $F \leftarrow \bigwedge\limits_{h \in H} h \wedge \bigwedge\limits_{\substack{s_i \in S, \\ i=1..|S|}} (s_i \vee p_i)$, где p_i — пропозициональная переменная $\forall i = 1..|S|$. Алгоритм 1 продолжает работать до тех пор, пока формула F выполнима, обновляя в процессе решения лучшую модель $bestModel$, найденную к текущему моменту времени. Пропозициональные переменные $p_i : i = 1..|S|$ используются для “отката” из уже просмотренных состояний с помощью добавления дополнительных утверждений (через

Алгоритм 1 WMax

Input: H, S, W

1: $F \leftarrow \bigwedge\limits_{h \in H} h \wedge \bigwedge\limits_{\substack{s_i \in S, \\ i=1..|S|}} (s_i \vee p_i)$

2: ▷ p_i — пропозициональная переменная

3: $cost \leftarrow 0,\ minCost \leftarrow$ **null**

4: $isSat \leftarrow$ **true**, $model \leftarrow$ **null**, $bestModel \leftarrow$ **null**

5:

6: **while** $isSat$ **do**

7: $\quad isSat \leftarrow$ SOLVESAT(F)

8:

9: $\quad$ **if** $isSat \neq$ **true then**

10: $\quad\quad$ **return** $bestModel$

11: $\quad$ **end if**

12:

13: $\quad model \leftarrow$ GETMODEL()

14:

15: $\quad$ **if** $\forall p_i,\ i = 1..|S|\ p_i =$ **false then**

16: $\quad\quad bestModel \leftarrow model$

17: $\quad\quad$ **return** $bestModel$

18: $\quad$ **end if**

19:

20: $\quad$ **if** $\exists p_i,\ i = 1..|S| :$ EVAL($model$, p_i) = **true then**

21: $\quad\quad cost \leftarrow \sum\limits_{\substack{i=1..|S|, \\ p_i = \textbf{true}}} weight(s_i)$

22: $\quad$ **end if**

23:

24: $\quad$ **if null** $\neq minCost <= cost$ **then**

25: $\quad\quad F \leftarrow F \wedge \left(\bigvee\limits_{\substack{i=1..|S|, \\ p_i = \textbf{true}}} \neg p_i \right)$

26: $\quad$ **end if**

27:

28: $\quad$ **if** $minCost =$ **null or** $cost < minCost$ **then**

29: $\quad\quad F \leftarrow F \wedge \left(\bigvee\limits_{\substack{i=1..|S|, \\ p_i = \textbf{true}}} \neg p_i \right)$

30: $\quad\quad bestModel \leftarrow model$

31: $\quad$ **end if**

32: **end while**

конъюнкцию) в формулу F.

Существенный недостаток подхода заключается в медленной сходимости. Самой "дорогой" операцией при решении $MaxSMT$ является вызов SMT-решателя. Оценим количество запросов к SMT-решателю в процессе решения задачи.

Теорема 1. *Алгоритм 1 в худшем случае делает $2^{|S|}$ запросов к SMT-решателю.*

Доказательство. Пусть на каждой итерации алгоритм сопоставляет значениям пропозициональных переменных точку в дискретном $|S|$-мерном пространстве (биекция). Для $|S|$ исходных утверждений существует $2^{|S|}$ точек. По построению алгоритма нельзя прийти в одну и ту же точку повторно, так как алгоритм вычеркивает из рассмотрения пройденные точки (строки 25 и 29). В худшем случае, до того как алгоритм завершится, будут просмотрены все точки.

Худший случай достигается, когда все утверждения из S выполнимы, а точки просматриваются в таком порядке:

1. все пропозициональные переменные истинны;
2. все, кроме одной пропозициональной переменной истинны;
3. все, кроме двух пропозициональных переменных истинны;
4. и т.д.

Тогда будет сделано $C^{|S|}_{|S|} + C^{|S-1|}_{|S|} + ... + C^{2}_{|S|} + C^{1}_{|S|} + C^{0}_{|S|} = \sum\limits_{k=0}^{|S|} C^{k}_{|S|} = \sum\limits_{k=0}^{|S|} \frac{|S|!}{k! * (|S| - k)!}$ запросов к SMT-решателю, что по свойству суммы биномиальных коэффициентов равно $2^{|S|}$. □

По Теореме 1 в худшем случае алгоритм WMax делает экспоненциальное от $|S|$ число запросов к решателю.

Основное преимущество подхода — достижение субоптимальности минимальными усилиями. Достаточно в момент завершения выделенного лимита времени вернуть текущее значение, сохраненное в $bestModel$.

1.3.2 Подход к решению задачи $MaxSMT$, основанный на использовании ядер

Алгоритм 2 описывает подход к решению задачи $MaxSMT$, основанный на использовании ядер в качестве основного инструмента достижения цели.

Сначала формируется набор утверждений $F \leftarrow H \cup S$ (строка 1). Алгоритм 2 работает до тех пор, пока формула $\bigwedge_{f \in F} f$ невыполнима. Итерация алгоритма состоит из следующих шагов:

1. проверки формулы на выполнимость (строка 5);
2. возвращении модели (оптимального решения) в случае выполнимости формулы (строки 7-9);
3. минимальном "ослаблении" ядра (причем с учетом весов) в случае невыполнимости формулы (строка 12).

Алгоритм 2 Подход, основанный на использовании ядер

Input: H, S

```
1:  F ← H ∪ S
2:  isSAT ← false
3:
4:  while isSAT = false do
5:      isSAT = checkSAT()
6:
7:      if isSAT = true then
8:          return getModel()
9:      end if
10:
11:     core ← getUnsatCore()              ▷ только утверждения из S
12:     F ← (F ∖ core) ∪ minRelax(core)
13: end while
```

Минимальное "ослабление" ядра за конечное число шагов приводит Алгоритм 2 к корректному завершению (при условии, что формула F из разрешимой теории или комбинации теорий).

Например, если набор утверждений $\{x_1, x_2, x_3\}$ с единичными (для простоты) весами образует ядро, то функция вида $f(x, y, z) = (\neg x \vee \neg y \vee \neg z) \wedge ((x \wedge y) \vee (x \wedge z) \vee (y \wedge z))$ осуществляет минимальное "ослабление" такого ядра. Она делает в точности следующее: позволяет выполниться любым двум утверждениям (ведь три утверждения уже образуют ядро).

Существует несколько способов достижения минимального "ослабления" ядра:

- использование ограничений на кардинальность;
- использование правила вывода максимальной резолюции (MaxRes).

Ограничение на кардинальность на множество утверждений K вида $card(K) \leq n$ (аналогично для $\geq$), $n \in \mathbb{N}$ обозначает, что максимум n утверждений может выполняться одновременно.

Алгоритм OLL [8], судя по результатам соревнований MaxSAT Evaluation 2023[3], является наиболее эффективным алгоритмом, построенным на использовании ядер. Минимальное "ослабление" ядер в OLL достигается путем использования ограничений на кардинальность. Однако ограничения на кардинальность поддерживаются далеко не всеми решателями. Разбатываемый $MaxSMT$-решатель должен поддерживать возможность работы с разными решателями, из-за чего алгоритмы, построенные с использованием ограничений на кардинальность, не подходят для реализации.

Более универсальными являются алгоритмы решения задачи $MaxSMT$, использующие в своей реализации правило вывода максимальной резолюции. В таком случае достаточно, чтобы решатель поддерживал извлечение ядер.

MaxRes позволяет "ослабить" набор утверждений, образующих ядро, выдавая взамен новые утверждения, добавляемые в множества H и S по резолютивному правилу вывода [9].

[3]https://maxsat-evaluations.github.io/2023/ (дата обращения: 24.12.2023)

Использование максимальной резолюции для решения задачи $MaxSMT$ стало эффективным с появлением алгоритма PMRes [9]. Основное достижение алгоритма — использование "сжатой" версии правила вывода максимальной резолюции, что позволяет избежать квадратичного "взрыва" формулы для худших случаев. Алгоритм PMRes до сих пор является довольно эффективным согласно результатам соревнований MaxSAT Evaluation за последние несколько лет.

Важным преимуществом подхода, основанного на использовании ядер в качестве сущностей, направляющих к решению, является эффективность. Докажем это, показав, что в случае применения такого подхода в худшем случае будет сделано $|S|$ обращений к SMT-решателю против $2^{|S|}$ для подхода, основанного на итеративном улучшении модели.

Теорема 2. *Алгоритм 2 в худшем случае делает $|S|$ запросов к SMT-решателю.*

Доказательство. Пусть все утверждения из S невыполнимы (очевидно, что чем больше утверждений невыполнимо, тем больше итераций "ослаблений" и, соответственно, вызовов к SMT-решателю будет сделано). Тогда ослабим формулу $F \leftarrow \bigwedge_{h \in H} h \wedge \bigwedge_{s \in S} s$ так, что позволим выполняться ровно одному утверждению из S. Если после этого шага формула все еще невыполнима, то снова ослабим ее, позволив выполняться еще одному утверждению из S. Процесс будем повторять до тех пор, пока формула F не станет выполнимой. На каждой итерации алгоритма делается один запрос к SMT-решателю, к концу работы алгоритма будет сделано $|S|$ запросов к решателю. □

Однако Алгоритм 2 обладает существенным недостатком: он лишен субоптимальности, так как не предоставляет промежуточные субоптимальные решения.

Тем не менее, существуют модификации Алгоритма 2, позволяющие получать промежуточные модели. Например, алгоритм PrimalDualMaxRes [1].

1.3.3 Подход к решению задачи $MaxSMT$, основанный на комбинации использования ядер и получения моделей

Алгоритм PrimalDualMaxRes является алгоритмом решения задачи $MaxSAT$. В отличие от других рассмотренных алгоритмов он требует внесения изменений для того, чтобы использоваться для задачи $MaxSMT$.

Помимо понятия ядра алгоритм оперирует таким термином как корректирующий набор — .

Для простоты повествования будем считать, что все утверждения с весами оснащены единичными весами. (Написать про то как перейти к ПРОИЗВОЛЬНЫМ весам, описано дальше в статье — cloning).

Алгоритм в процессе работы обновляет два значения:

- значение верхней границы lb, изначально равное 0;
- значение нижней границы ub, изначально равное $\sum_{s \in S} weight(s)$.

Работает алгоритм до тех пор, пока значение нижней границы меньше верхней.

Основная идея алгоритма заключается в трех действиях:

- фазе разделения;
- проверки на выполнимость утверждений без весов с учетом результата работы фазы разделения;
- преобразовании полученного ядра или корректирующего набора с соответствующим обновлением значений границ.

1.3.3.1 Фаза разделения

В фазу разделения обрабатываются утверждения с весами.

- Берется утверждение, значение которого еще не установлено. Его значение устанавливается в $True$.

Алгоритм 3 PrimalDualMaxRes

Input: $F = \{ (b_1, 1), ..., (b_m, 1), C_{m+1}, C_n \}$

$i \leftarrow 0, F^0 \leftarrow F, ub \leftarrow m, lb \leftarrow 0$

$I^* = \{ b_1 = 0, ..., b_m = 0 \}$ ▷ Текущее наилучшее решение

solver.add(hardCls(F_0)) ▷ Add hard clauses to a SAT solver

while lb $<$ ub **do**

  Undo all branches, reset I

  { Фаза "разделения" }

  for $(b_1) \in F_i$ **do**

    if $I(b)$ не определена **then**

      solver.branch($b = 1$) ▷ обновляет $I(b) = 1$

      solver.propagate() ▷ обновляет I, выводя значения

    end if

    $\varphi_i = \{(b_1) \in F_i \mid S_I(b) = 1\}$ ▷ Кандидат на ядро

    $\pi_i = \{(b_1) \in F_i \mid S_I(b) = 0\}$ ▷ Кандидат на корректирующий набор

    (issat, φ_i) = solver.solve(I)

  end for

  { Фаза "трансформации" }

  if (issat $\wedge$ $\pi_i = \emptyset$) **then return** $(I,$ lb$)$

  else if ($\neg$issat $\wedge$ $\varphi_i = \emptyset$) **then return** $(I^*,$ ub$)$

  else if (isssat $\wedge$ $\pi_i \neq \emptyset$) **then**

    F_{i+1} = TransformCSet(F_i, π_i)

    if maxcost(π_i) $<$ ub $-$ lb **then**

      $I^* = I$

      ub = maxcost(π_i) + lb

    end if

  else if ($\neg$isssat $\wedge$ $\varphi_i \neq \emptyset$) **then**

    F_{i+1} = TransformCore(F_i, φ_i)

    lb = lb + 1

    $i = i + 1$

    solver.add(hardCls(F_{i+1}) $\setminus$ hardCls(F_i)) ▷ Add new hard clauses

  end if

end while

return $(I^*,$ ub$)$

- По значению утверждения выводятся значения пропозициональных переменных, входящих в утверждений, а так же значения некоторых других утверждений (англ. unit propagation).

Результатом работы I фазы разделения является набор пропозициональных переменных, входящих в утверждения с весами, и сопоставленные им значения.

Отметим, что такой процесс порождает два непересекающихся множества:

- потенциальное ядро φ: $\{ s \in S \mid s = True \}$;
- потенциальный корректирующий набор π: $\{ s \in S \mid s = False \}$.

Так же заметим, что этот шаг возможно выполнить, работая с пропозициональными переменными, но он не расширяется на произвольные теории, а значит требует адаптации для работы с ними.

1.3.3.2 Проверка на выполнимость утверждений без весов с учетом результата работы фазы разделения

Далее SAT-решатель проверяет на выполнимость $H|_I$ — утверждения без весов с учетом результата, полученного в фазу разделения.

Проверка $H|_I$ на выполнимость приведет к одной из возможных комбинаций:

1. формула выполнима, при этом $\pi = \emptyset$;
2. формула невыполнима, при этом решатель вернул ядро $\varphi_0 = \emptyset$;
3. формула выполнима, при этом $\pi \neq \emptyset$;
4. формула невыполнима, при этом решатель вернул ядро $\varphi_0 \neq \emptyset$, $\varphi_0 \subseteq \varphi$.

В первом случае текущее I — оптимальная модель, позволяющая определить какие из утверждений с весами выполнимы. Во втором случае пустое ядро означает невыполнимость утверждений без весов, а

следовательно, отсутствие решения задачи $MaxSAT$. Третий случай означает, что оптимальное решение еще не найдено, ведь корректирующий набор не пуст. В таком случае стоит преобразовать формулу (это в первую очередь включает преобразование корректирующего набора), при этом обновив значение верхней границы. Четвертый случай так же означает, что оптимальное решение еще не было найдено. Он требует преобразования формулы (в частности ядра) и обновления уже нижней границы.

2 Вывод

Были исследованы существующие подходы к решению задачи $MaxSAT$: на основе получения моделей, с использованием ядер, комбинирование получения моделей и использования ядер. Выбран третий подход, потому что он предлагает линейную сходимость и может выдавать субоптимальные решения (первый подход имеет экспоненциальную сходимость, второй не выдает субоптимальных решений). В рамках третьего подхода был выбран алгоритм PrimalDualMaxRes.

3 Адаптация выбранного алгоритма под решение задачи $MaxSMT$

Как было отмечено ранее, в алгоритме PrimalDualMaxRes фаза "разделения" не обобщается с решения задачи $MaxSAT$ на решение задачи $MaxSMT$. Замена этого шага на аналогичный по смыслу, но адаптированный под работу с произвольными теориями, может адаптировать алгоритм под решение задачи $MaxSMT$. Для того, чтобы это сделать, необходимо понять в чем заключается смысл шага. Основная идея фазы "разделения" заключается в получении промежуточной модели, то есть промежуточного решения. Далее проверка формулы на выполнимость осуществляется уже с учетом полученного промежуточного решения. То есть необходимо получить промежуточное решение, но для задачи $MaxSMT$. Будем делать это следующим образом. Пусть есть множества H и $S = \{s_1, s_2, ..., s_n\}$, множества утверждений без весов и с весами соответственно в терминах задачи $MaxSMT$. Будем на первом шаге проверять на выполнимость такую формулу: $F_1 \leftarrow \bigwedge\limits_{h \in H} h \wedge (s_1 \wedge s_2)$. В случае если формула выполнима, промежуточное решение найдено. В противном случае будем проверять на выполнимость формулу F_2 — конъюнкцию формулы F_1 с несколькими новыми ограничениями из множества S. Очевидно, что в случае выполнимости формулы, будет получено промежуточное решение. В случае невыполнимости формулы шаги можно будет

$$F \leftarrow \bigwedge_{h \in H} h \wedge \bigwedge_{\substack{s_i \in S, \\ i=1..|S|}} (s_i)$$

4 Реализация

4.1 Прикладной программный интерфейс, предоставляемый MaxSMT-решателем

Основные возможности, которые предоставляет $MaxSMT$-решатель (Рис. 2):

- добавить ограничение, которое должно выполняться;
- добавить ограничение с весом;
- попытаться решить задачу $MaxSMT$, ожидая получить оптимальное решение;
- попытаться решить задачу $MaxSMT$, ожидая получить субоптимальное решение.

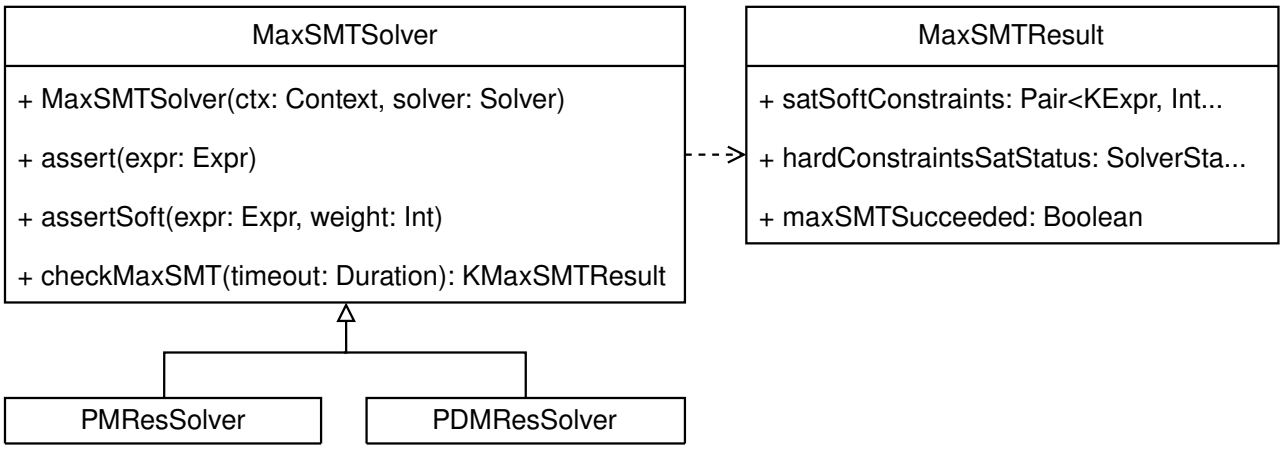

Рис. 1: Диаграмма классов $MaxSMT$-решателя (НУЖНО ОБНОВИТЬ)

В качестве результата решения задачи $MaxSMT$ будет возвращен объект класса KMaxSMTResult. Он содержит следующие свойства:

- значение, показывающее, выполнимы ли утверждения, выполнимость которых равносильна наличию решения задачи (как в задаче SMT);
- список выполнимых ограничений с весами (из S);

- значение, показывающее, успешно ли выполнена задача.

Решатель основан на алгоритме PrimalDualMaxRes и поддерживает различные конфигурации запуска. Конкретная конфигурации настраивается через объект класса KMaxSMTContext. Конфигурация состоит из набора опций, которые могут быть включены или отключены. Одна из существующих опций позволяет минимизировать ядра перед дальнейшей работой с ними. Другая — позволяет отдавать предпочтения ядрам с бóльшими весами. Третья позволяет извлекать вместо одного ядра несколько непересекающихся ядер и применять дальнейшие шаги к ним. (ПОЯСНИТЬ за смысл этой опций)

4.2 Взаимодействие $MaxSMT$-решателя с нативным SMT-решателем

Рис. 1 показывает как взаимодействует $MaxSMT$-решатель с нативными решателями. Например, программный интерфейс, реализованный поверх решателя Z3, позволяет вызывать решатель из JVM-кода. В KSMT реализован конвертер, позволяющий переводить формулы из представления для решателя Z3 на языке Java в представление формул библиотеки KSMT. Именно с этим представлением формул и работает $MaxSMT$-решатель.

4.3 MaxSMT-решатель в режиме портфолио

Этот раздел будет описан после анализа результатов, в зависимости от них реализация может измениться или в нее могут быть внесены изменения!!!

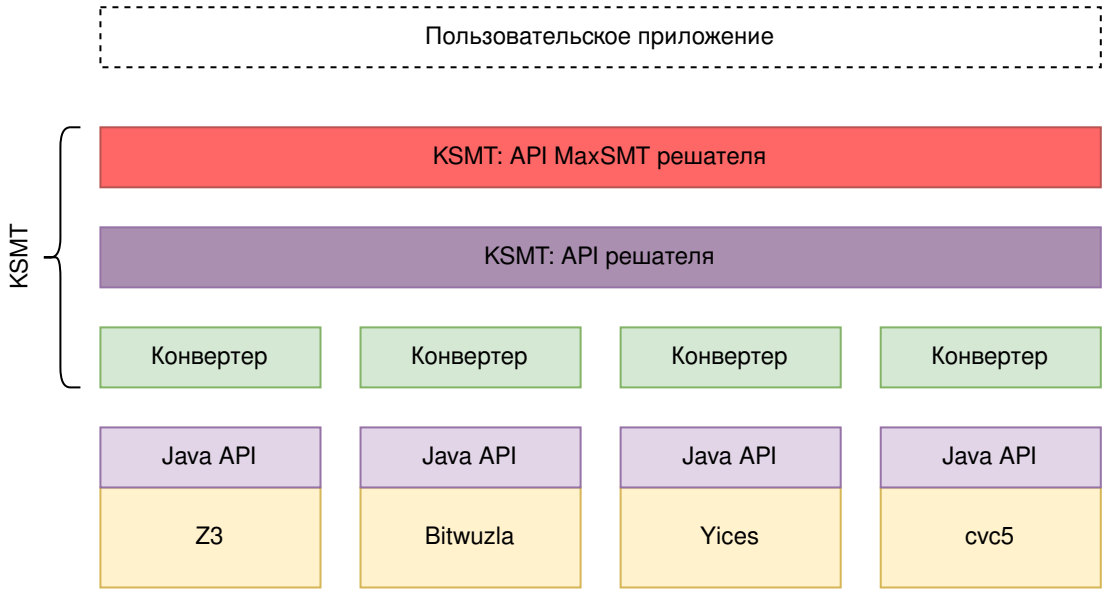

Рис. 2: Диаграмма взаимодействия $MaxSMT$-решателя с нативным SMT-решателем

5 Тестирование

Для оценки корректности реализованных алгоритмов был настроен прогон на тестовом наборе данных для задачи $MaxSAT$[4].

Также был создан тестовый набор данных для задачи $MaxSMT$ для бескванторных фрагментов таких теорий как: теории массивов, битовых векторов, неинтерпретированных функций, чисел с плавающей точкой, линейной целочисленной и рациональной арифметик, а так же различных комбинаций этих теорий.

Тестовый набор данных создан следующим образом:

1. преобразован тестовый набор данных для задачи SMT[5];
2. с помощью $MaxSMT$-решателя νZ (часть Z3) нагенерированы ответы на задачу $MaxSMT$.

Корректность алгоритмов была проверена на обоих наборах тестовых данных.

[4]https://maxsat-evaluations.github.io/2023/ (дата обращения: 04.05.2024)
[5]https://smtlib.cs.uiowa.edu/benchmarks.shtml (дата обращения: 04.05.2024)

Заключение

Код решения доступен в репозитории[6] GitHub.

Код скриптов, написанных для преобразования тестового набора данных для задачи SMT в задачу MaxSMT, доступен в репозитории[7] GitHub.

Код, запускающий MaxSMT-решатель νZ для генерации решения задачи $MaxSMT$, доступен в репозитории[8] GitHub.

Далее представлены выполненные задачи с кратким пояснением результатов.

1. Были исследованы существующие подходы к решению задачи MaxSAT: на получении моделей, на использовании ядер, на комбинации получения моделей и использования ядер. Первый подход дает экспоненциальную сходимость, второй — отсутствие субоптимальных решений, третий (комбинация) предлагает линейную сходимость с возможностью поиска субоптимальных решений.

2. Был выбран алгоритм для решения задачи MaxSAT с возможностью поиска субоптимальных решений, который можно адаптировать под решение MaxSMT. В нем заменен шаг распространения значения пропозициональной переменной на аналогичный по смыслу в контексте алгоритма, но для работы с произвольными теориями.

3. Адаптированный алгоритм реализован в библиотеке KSMT. Реализация поддерживает режим портфолио.

4. Корректность реализованного алгоритма была оценена на основе существующего тестового набора данных для MaxSAT, а также с помощью сгенерированного тестового набора данных, получен-

[6]https://github.com/UnitTestBot/ksmt/pull/137 (дата обращения: 04.05.2024)

[7]https://github.com/viktoriia-fomina/MaxSMT-benchmark-generation (дата обращения: 04.05.2024)

[8]https://github.com/viktoriia-fomina/ksmt-maxsmt-gen (дата обращения: 04.05.2024)

ного путем преобразования тестового набора данных для задачи SMT.

5. (Сравнить скорость работы MaxSMT-решателя в режиме портфолио и без. — это осталось сделать)

Список литературы

[1] Bjorner Nikolaj, Narodytska Nina. Maximum satisfiability using cores and correction sets // Twenty-Fourth International Joint Conference on Artificial Intelligence. — 2015.

[2] Bjørner Nikolaj, Phan Anh-Dung, Fleckenstein Lars. νz-an optimizing SMT solver // Tools and Algorithms for the Construction and Analysis of Systems: 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015, Proceedings 21 / Springer. — 2015. — P. 194–199.

[3] Bjørner Nikolaj S, Phan Anh-Dung. νZ-Maximal Satisfaction with Z3. // Scss. — 2014. — Vol. 30. — P. 1–9.

[4] Candea George, Godefroid Patrice. Automated software test generation: some challenges, solutions, and recent advances // Computing and Software Science: State of the Art and Perspectives. — 2019. — P. 505–531.

[5] Compositional safety verification with Max-SMT / Marc Brockschmidt, Daniel Larra, Albert Oliveras et al. // 2015 Formal Methods in Computer-Aided Design (FMCAD) / IEEE. — 2015. — P. 33–40.

[6] De Moura Leonardo, Bjørner Nikolaj. Z3: An efficient SMT solver // International conference on Tools and Algorithms for the Construction and Analysis of Systems / Springer. — 2008. — P. 337–340.

[7] Inverso Omar, Trubiani Catia. Parallel and distributed bounded model checking of multi-threaded programs // Proceedings of the 25th ACM SIGPLAN symposium on principles and practice of parallel programming. — 2020. — P. 202–216.

[8] Morgado António, Dodaro Carmine, Marques-Silva Joao. Core-guided MaxSAT with soft cardinality constraints // International Conference

on Principles and Practice of Constraint Programming / Springer. — 2014. — P. 564–573.

[9] Narodytska Nina, Bacchus Fahiem. Maximum satisfiability using core-guided MaxSAT resolution // Proceedings of the AAAI Conference on Artificial Intelligence. — Vol. 28. — 2014.

[10] Optimizing symbolic execution for malware behavior classification / Stefano Sebastio, Eduard Baranov, Fabrizio Biondi et al. // Computers & Security. — 2020. — Vol. 93. — P. 101775.

[11] Pasareanu Corina S, Phan Quoc-Sang, Malacaria Pasquale. Multi-run side-channel analysis using Symbolic Execution and Max-SMT // 2016 IEEE 29th Computer Security Foundations Symposium (CSF) / IEEE. — 2016. — P. 387–400.

[12] Proving non-termination using Max-SMT / Daniel Larraz, Kaustubh Nimkar, Albert Oliveras et al. // Computer Aided Verification: 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings 26 / Springer. — 2014. — P. 779–796.

[13] Proving termination of imperative programs using Max-SMT / Daniel Larraz, Albert Oliveras, Enric Rodríguez-Carbonell, Albert Rubio // 2013 Formal Methods in Computer-Aided Design / IEEE. — 2013. — P. 218–225.

[14] Satisfiability modulo theories / Clark Barrett, Roberto Sebastiani, Sanjit A Seshia, Cesare Tinelli. — 2021.

[15] Sebastiani Roberto, Trentin Patrick. On optimization modulo theories, MaxSMT and sorting networks // International Conference on Tools and Algorithms for the Construction and Analysis of Systems / Springer. — 2017. — P. 231–248.

[16] Sebastiani Roberto, Trentin Patrick. OptiMathSAT: A tool for optimization modulo theories // Journal of Automated Reasoning. — 2020. — Vol. 64, no. 3. — P. 423–460.

[17] Synthesis of super-optimized smart contracts using max-smt / Elvira Albert, Pablo Gordillo, Albert Rubio, Maria A Schett // International Conference on Computer Aided Verification / Springer. — 2020. — P. 177–200.

[18] A comprehensive study and analysis on SAT-solvers: advances, usages and achievements / Sahel Alouneh, Sa'ed Abed, Mohammad H Al Shayeji, Raed Mesleh // Artificial Intelligence Review. — 2019. — Vol. 52. — P. 2575–2601.

[19] The mathsat5 smt solver / Alessandro Cimatti, Alberto Griggio, Bastiaan Joost Schaafsma, Roberto Sebastiani // International Conference on Tools and Algorithms for the Construction and Analysis of Systems / Springer. — 2013. — P. 93–107.