Санкт-Петербургский государственный университет

Системное программирование

Группа 22.Б07-мм

Разработка MaxSMT решателя

Фомина Виктория Викторовна

Отчёт по учебной практике
в форме «Решение»

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

Консультант:
преподаватель, ИПКН, ИТМО В. Соболь

Санкт-Петербург
2023

Оглавление

Введение

Во многих сферах профессиональной и научной жизни приходится решать задачи оптимизации.

Это, например, задача логистики — нахождение наименее ресурсоемкого плана. Или задача вероятностного вывода (англ. probabilistic inference) в статистике [7] — поиск наиболее вероятного объяснения. Задача обновления пакетов в ОС Linux [8] — поиск способа обновить наибольшее количество пакетов. И даже такие такие задачи как отладка параллельного кода [3], вывод типов в Python 3 [5] можно представить как задачи оптимизации.

Ясно, что существует необходимость в решении задач оптимизации формальными методами (потому что их много, и они везде встречаются). Оказывается, что многие задачи оптимизации можно представить в теориях или комбинациях теорий логики первого порядка [7, 5, 8, 4, 3].

Такие задачи можно закодировать в задачу $MaxSMT$ — оптимизационную версию задачи выполнимости в теории (задача SMT).

Задачу $MaxSMT$ можно поставить так:

- имеется набор утверждений H, выполнимость которых равносильна наличию решения задачи (как в задаче SMT);
- имеется набор утверждений S, которые могут не выполняться. При этом каждому утверждению из этого набора соответствует какой-то приоритет его выполнения — вес.

Важно понимать, что задача $MaxSMT$ — сложная вычислительная задача.

- решение задачи $MaxSMT$ требует многократно решать задачу SMT.
- сама задача SMT тоже имеет высокую вычислительную сложность и, более того, разрешима далеко не для всех теорий. Такие ограничения естественным образом приводят к тому, что SMT

решатели (далее решатели) обычно используют с ограничением по времени.

Обоснованно использовать $MaxSMT$ решатели с ограничением по времени и в таком случае искать не оптимальное, а субоптимальное решение.

Также обоснованно для $MaxSMT$ решателя иметь возможность переключения между SMT решателями. Ведь SMT решатели показывают разную производительность даже на одинаковых наборах входных данных[1]. Естественно желание использовать результат решения того решателя, который может справится с задачей быстрее.

Написание решателя задачи $MaxSMT$ с возможностью поиска субоптимальных решений, поддерживающего переключение в между SMT решателями, и стало целью этой работы.

[1]SMT-COMP 2023 — https://smt-comp.github.io/2023/ (дата обращения: 10.10.2023)

1 Постановка задачи

Целью работы является разработка $MaxSMT$ решателя с возможностью поиска субоптимальных решений.

Для её выполнения были поставлены следующие задачи:

1. изучить существующие алгоритмы, решающие задачу $MaxSMT$;
2. реализовать в KSMT один или несколько алгоритмов, решающих задачу $MaxSMT$;
3. расширить существующий тестовый набор данных для различных теорий, оценивающий корректность решения задачи $MaxSMT$;
4. оценить корректность реализации алгоритмов на основе расширенного тестового набора данных;
5. исследовать возможность поиска субоптимальных решений с помощью рассмотренных алгоритмов;
6. разработать субоптимальный решатель задачи $MaxSMT$.

2 Обзор

2.1 Определения и нотации, используемые в работе

Перечислим основные нотации, используемые в работе:

- Формула φ логики первого порядка выполнима будем записать как: формула φ SAT (англ. satisfiable).
- Формула φ логики первого порядка невыполнима будем записать как: формула φ UNSAT (англ. unsatisfiable).
- Модель – отображение $M : V \to \mathbb{B}$, где V – множество предметных переменных, а $\mathbb{B}$ – множество истинностных значений.
- UNSAT ядро (или обоснование) – подмножество множества мягких ограничений (S), конъюнкция которых с ограничениями из множества жёстких ограничений (H) невыполнима.

2.2 Обзор алгоритмов решения задачи $MaxSMT$

Существует два основных подхода к решению задачи $MaxSMT$:

- основанный на итеративном улучшении моделей;
- основанный на получении UNSAT ядер (обоснований).

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

Идея решения задачи с помощью получения моделей состоит в получении модели и итеративном улучшении этой модели. Одним из алгоритмов, реализующим этот подход, является алгоритм WMax [2]. Основной недостаток такого подхода заключается в медленной сходимости, что доказывают различные статьи и соревнования SMT решателей, среди участников и победителей которых за последний годы не существует решателей на основе таких алгоритмов.

2.2.2 Подход к решению задачи $MaxSMT$, основанный на получении UNSAT ядер

Сформулируем идею решения задачи $MaxSMT$ с использованием UNSAT ядер.

Пусть формула φ — конъюнкция ограничений из множества H и S. Тогда будем выполнять следующие шаги.

1. Проверяем, что конъюнкция ограничений из множества H SAT.
    - Если она UNSAT, то задача $MaxSMT$ не имеет решения — конец.
2. Повторяем пока формула φ UNSAT.
    (a) Проверяем формулу φ на выполнимость.
        - Если она SAT — конец. Найдено решение задачи $MaxSMT$.

    (b) Ослабляем формулу φ таким образом, что больше мягких ограничений могут быть невыполнимыми.
        - Минимальное ослабление приведет к тому, что будет найдено решение задачи $MaxSMT$, если формула φ SAT.

Существует несколько подходов к получению минимального ослабления:

- использование ограничений на кардинальность;
- использование максимальной резолюции.

OLL судя по результатам соревнований MaxSAT Evaluation 2023[2] является наиболее эффективным алгоритмом, использующим в своей реализации ограничения на кардинальность. Тем не менее, ограничения на кардинальность поддерживаются далеко не во всех решателях, поэтому такой алгоритм не подходит для реализции MaxSMT решателя в KSMT.

[2] https://maxsat-evaluations.github.io/2023/ (дата обращения: 10.11.2023)

Более универсальными являются MaxSMT решатели, использующие метод максимальной резолюции.

Метод максимальной резолюции – это правило вывода, которое как и обычный метод резолюции, берет две формулы и выводит из них новую. Однако в отличие от обычного метода резолюции, метод максимальной резолюции:

- удаляет исходные формулы;
- выводит множество новых формул.

Первым таким алгоритмом, и до сих пор эффективным согласно результатам соревнований MaxSAT Evaluation за последние несколько лет, является алгоритм PMRes [6] и его модификации [1].

3 Реализация

3.1 Взаимодействие MaxSMT решателя с нативным SMT решателем

Для реализации алгоритма $MaxSMT$ была выбрана Kotlin-библиотека KSMT[3] с открытым исходным кодом. Библиотека KSMT предоставляет унифицированный API к нескольким решателям и позволяет переключаться между ними в процессе решения задачи SMT, выбирая к качестве результата решение того решателя, который быстрее других справился с задачей (режим портфолио). Режим портфолио позволяет автоматически поддержать переключение между SMT решателями в процессе решения задачи $MaxSMT$.

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.

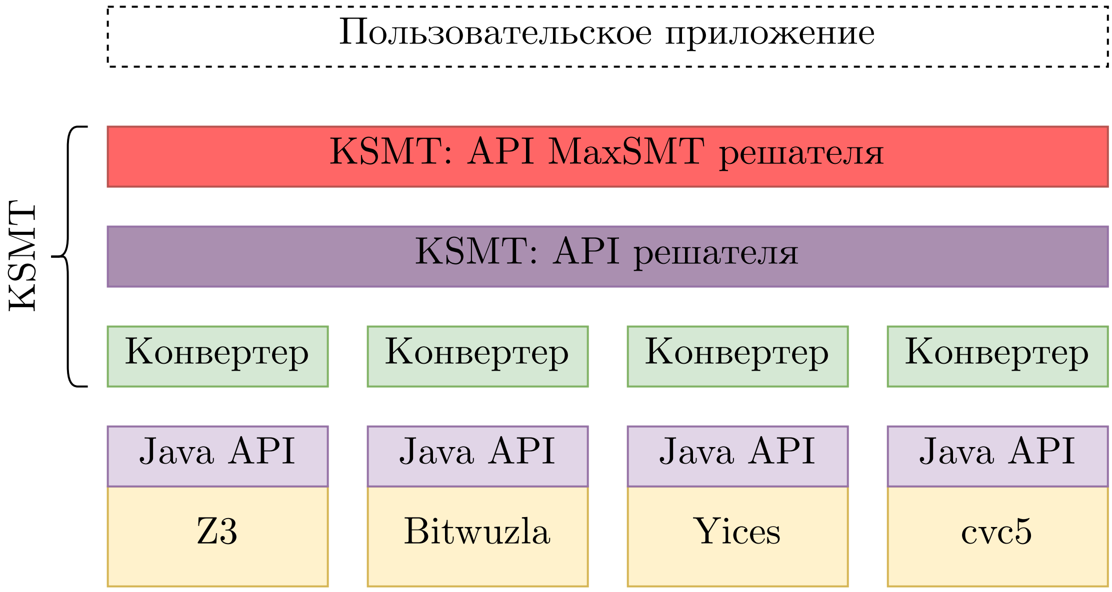

Рис. 1: Диаграмма взаимодействия $MaxSMT$ решателя с нативным SMT решателем

[3] https://ksmt.io/ (дата обращения: 10.10.2023)

3.2 API $MaxSMT$ решателя и реализованные решатели

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

- добавить ограничение, которое должно выполняться;
- добавить ограничение с весом;
- попытаться решить задачу $MaxSMT$.

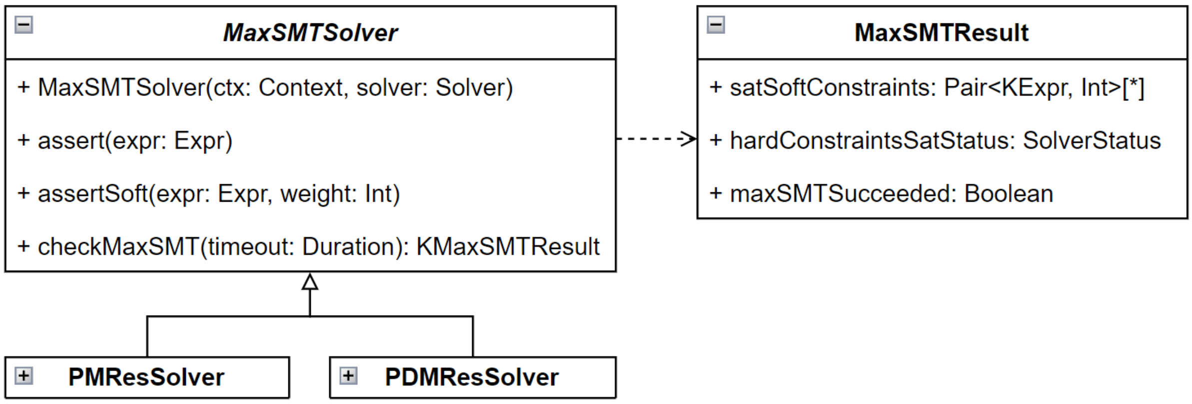

Рис. 2: Диаграмма классов $MaxSMT$ решателя

3.3 Расширение тестового набора данных, оценивающего корректность решения задачи MaxSMT

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

Было реализовано два $MaxSMT$ решателя: на основе алгоритмов PMResSolver и PDMResSolver.

Также был создан тестовый набор данных для задачи $MaxSMT$ для бескванторных фрагментов таких теорий как: теории массивов, битовых векторов, неинтерпретированных функций, чисел с плавающей

[4]https://maxsat-evaluations.github.io/2023/ (дата обращения: 10.10.2023)

точкой, линейной целочисленной и рациональной арифметик (а так же различных комбинаций этих теорий).

Тестовый набор данных был создан следующим образом:

1. был преобразованный тестовый набор данных для задачи SMT[5];

2. был запущен $MaxSMT$ решатель νZ (часть Z3).

[5]https://smtlib.cs.uiowa.edu/benchmarks.shtml (дата обращения: 10.10.2023)

4 Тестирование

Корректность реализованных алгоритмов была оценена на тестовом наборе данных для задачи $MaxSAT$.

Заключение

В текущем семестре были выполнены следующие задачи: Для её выполнения были поставлены следующие задачи:

1. изучить существующие алгоритмы, решающие задачу $MaxSMT$;
2. реализовать в KSMT один или несколько алгоритмов, решающих задачу $MaxSMT$;
3. расширить существующий тестовый набор данных для различных теорий, оценивающий корректность решения задачи $MaxSMT$.

Следующие задачи планируется выполнить в последующие семестры:

1. оценить корректность реализации алгоритмов на основе расширенного тестового набора данных;
2. исследовать возможность поиска субоптимальных решений с помощью рассмотренных алгоритмов;
3. разработать субоптимальный решатель задачи $MaxSMT$.

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

[3] Concurrency debugging with MaxSMT / Miguel Terra-Neves, Nuno Machado, Inês Lynce, Vasco Manquinho // Proceedings of the AAAI Conference on Artificial Intelligence. — Vol. 33. — 2019. — P. 1608–1616.

[4] Guerra João, Lynce Inês. Reasoning over biological networks using maximum satisfiability // International conference on principles and practice of constraint programming / Springer. — 2012. — P. 941–956.

[5] MaxSMT-based type inference for Python 3 / Mostafa Hassan, Caterina Urban, Marco Eilers, Peter Müller // Computer Aided Verification: 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Part II 30 / Springer. — 2018. — P. 12–19.

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

[7] Park James D. Using weighted MAX-SAT engines to solve MPE // AAAI/IAAI. — 2002. — P. 682–687.

[8] Solving linux upgradeability problems using boolean optimization / Josep Argelich, Daniel Le Berre, Inês Lynce et al. // arXiv preprint arXiv:1007.1021. — 2010.