DOI: 10.15514/ISPRAS-2025-37(4)-10

MaxSMT-решатель, поддерживающий режим портфолио

[1] *В.В. Фомина, ORCID: 0009-0003-7132-722X <viky.fomina@yandex.ru>*
[2] *В. Соболь, ORCID: 0000-0002-2371-225X <vsobol@itmo.ru>*
[1] *Д.В. Кознов, ORCID: 0000-0003-2632-3193 <d.koznov@spbu.ru>*

[1] *Санкт-Петербургский государственный университет,*
Россия, 199034, г. Санкт-Петербург, Университетская наб., д. 7–9.

[2] *Университет ИТМО,*
Россия, 197101, г. Санкт-Петербург, Кронверкский пр., д. 49, лит. А.

Аннотация. Задача MaxSMT заключается в определении выполнимости формулы теории первого порядка с ограничениями. В статье представлен подход к решению этой задачи для случая бесквантворных теорий, которые обычно разрешимы и востребованы на практике для генерации тестов к программному обеспечению. Подход включает в себя модифицированный MaxSAT-алгоритм PrimalDualMaxRes, в котором использование SAT-решателя заменено SMT-решателем, а также реализацию режима портфолио. Последний означает, что итоговый MaxSMT-алгоритм запускается параллельно с несколькими SMT-решателями, и в качестве конечного результата берется наилучший. В качестве SMT-решателей, на базе которых построен режим портфолио, были выбраны Z3, Yices и Bitwuzla. Подход реализован с использованием открытой Kotlin-библиотеки KSMT, реализующей инфраструктуру для работы с набором MaxSMT-решателей. В статье также представлен первый тестовый набор (benchmark) для MaxSMT-задачи. Проведенные эксперименты подтверждают эффективность предложенного решения. Разработанное портфолио MaxSMT-решателей справляется с примерами из тестового набора данных более, чем в четыре раза быстрее существующего MaxSMT-решателя νZ (проект Z3).

Ключевые слова: задача MaxSMT; MaxSMT-решатель; библиотека KSMT; SMT-решатель; режим портфолио.

Для цитирования: Фомина В.В., Соболь В., Кознов Д.В. MaxSMT-решатель, поддерживающий режим портфолио. Труды ИСП РАН, том 37, вып. 4, часть 1, 2025 г., стр. 177–188. DOI: 10.15514/ISPRAS–2025–37(4)–10.

MaxSMT Solver with Portfolio Mode Support

[1] *V.V. Fomina ORCID: 0009-0003-7132-722X <viky.fomina@yandex.ru>*
[2] *V. Sobol ORCID: 0000-0002-2371-225X <vsobol@itmo.ru>*
[1] *D.V. Koznov ORCID: 0000-0003-2632-3193 <d.koznov@spbu.ru>*

[1] *Saint-Petersburg State University,*
7/9 Universitetskaya emb., St. Petersburg, 199034, Russia.

[2] *ITMO University,*
Kronverksky Pr. 49, bldg. A, St. Petersburg, 197101, Russia.

Abstract. The MaxSMT problem is to determine the satisfiability of a first-order theory formula with constraints. The paper presents an approach to solving this problem for the case of quantifier-free theories, which are usually decidable and widely applicable in practice for software test generation. The approach includes a modified MaxSAT algorithm PrimalDualMaxRes, in which the use of a SAT solver is replaced by an SMT solver, as well as the implementation of the portfolio mode. The latter means that the final MaxSMT algorithm is run in parallel with several MaxSMT solvers, and the best one is taken as the result. Z3, Yices and Bitwuzla were chosen as the SMT solvers on which the portfolio mode is built. The approach is implemented using the open-source Kotlin library KSMT, which implements the infrastructure for working with a set of MaxSMT solvers. The paper also presents the first benchmark for the MaxSMT problem. The evaluation demonstrates the competitiveness of the proposed solution. The developed portfolio of MaxSMT solvers outperforms the existing MaxSMT solver νZ (Z3 project), solving instances from the benchmark more than four times faster.

Keywords: MaxSMT; MaxSMT solver; KSMT; SMT solver; portfolio.

For citation: Fomina V.V., Sobol V., Koznov D.V. MaxSMT solver with portfolio mode support. Trudy ISP RAN/Proc. ISP RAS, vol. 37, issue 4, part 1, 2025. pp. 177-188 (in Russian). DOI: 10.15514/ISPRAS-2025-37(4)-10.

1. Введение

В области анализа программ широко известна задача SAT (Boolean Satisfiability Problem), определяющая выполнимость булевых формул [1-2]. Эффективное решение этой задачи, известное как SAT-революция, позволило значительно повысить точность и масштабируемость автоматической верификации и тестирования программного обеспечения [1]. Обобщением этой задачи на исчисление предикатов первого порядка является задача SMT (Satisfiability Modulo Theories) [3-4]. Она позволяет проверять выполнимость формул, описывающих более широкое множество свойств программ, чем булевы формулы. В свете этого создаются различные теории первого порядка для формального описания различных структур данных – массивов [5], битовых векторов [5] и пр., что позволяет формализовывать и автоматически проверять сложное поведение программ. Активно используются бескванторные фрагменты таких теорий как битовые вектора, неинтерпретированные функции, числа с плавающей точкой, линейная целочисленная и рациональная арифметики. Исключение утверждений с кванторами гарантирует разрешимость перечисленных теорий, что позволяет определять выполнимость или невыполнимость поведения программ. При этом сохраняется возможность описывать широкий диапазон свойств программ. Для автоматизации проверок этих свойств используются SMT-решатели – Z3 [6], Yices [7], Bitwuzla [8] и др.

При практическом использовании SMT-решателей часто возникает необходимость гибко управлять формулами, которые передаются в решатель [9]. В частности, важно иметь возможность задавать различные приоритеты для проверки отдельных частей формулы, например, в случае, когда требуется определить выполнимость не всей формулы. Такая

задача называется MaxSMT [9-14] (аналогичная задача для SAT называется MaxSAT [15]). Задача MaxSMT предполагает многократное решение задачи SMT.

Формально, задача MaxSMT описывается так. Пусть в рамках некоторой теории первого порядка имеется два набора утверждений H и S. Для набора H ставим задачу SMT. Далее каждому утверждению s из набора S ставим в соответствие приоритет его выполнимости – вес weight(s). Решить задачу MaxSMT означает найти набор утверждений K из всех выполнимых утверждений, входящих в S, для которого значение $\sum_{s \in K} weight(s)$ является наибольшим из возможных. Условием наличия решения задачи является выполнимость всех утверждений из H.

Задача MaxSMT может использоваться при автоматической генерации тестов с применением символьного исполнения [16-17]. Применение MaxSMT-решателя совместно с символьной машиной позволяет создавать тесты, которые удовлетворяют определенным условиям пути, т.е. попадают в нужное место программы. Кроме того, в таких тестах выполняются дополнительные ограничения, не противоречащие условиям пути. Например, можно задать диапазоны для размеров генерируемых массивов. Подобные ограничения, в частности, позволяют управлять размерами создаваемых в программе объектов, что помогает избегать чрезмерного использования памяти.

Существующие исследования, посвященные MaxSMT [18-21], не рассматривают практические подходы к повышению скорости решения задачи, в то время как такие подходы могут позволить повысить применимость MaxSMT-решателей, в том числе для анализа программ, который требует решения задачи MaxSMT. Последний можно ускорить применением режима портфолио. Этот режим предполагает одновременный запуск различных MaxSMT-решателей на одной задаче с целью использовать решение того решателя, который показал лучший результат за фиксированное время.

Данная работа посвящена эффективному решению задачи MaxSMT и рассматривает как теоретические, так и практические подходы к ее решению.

В работе используется известный алгоритм решения задачи MaxSAT – PrimalDualMaxRes [18], в котором шаг, позволяющий получать промежуточные решения для задачи MaxSAT, заменен шагом, позволяющим получать промежуточные решения для более общей задачи MaxSMT. Это обеспечивает возможность по истечении выделенного лимита времени вернуть наилучшее вычисленное решение (субоптимальное), даже если оно не является оптимальным. В качестве способа взаимодействия с SMT-решателями используется проект KSMT [22], который предоставляет универсальный интерфейс работы с различными SMT-решателями. Это позволяет реализовать решение задачи MaxSMT с использованием всех доступных в библиотеке SMT-решателей. В рамках работы создан тестовый набор данных (benchmark) для задачи MaxSMT [23].

Статья организована следующим образом. В разделе 2 обосновывается выбор технологий для решения поставленной задачи. В разделе 3 выполнен краткий обзор предложенного решения. В разделе 4 приводится адаптация алгоритма PrimalDualMaxRes для решения задачи MaxSMT. В разделе 5 предлагается реализация режима портфолио. В разделе 6 описываются эксперименты. В разделе 7 выполнен обзор близких к данному исследованию работ.

2. Выбор технологий

Известные алгоритмы решения задачи MaxSMT имеют ряд ограничений. Некоторые из них не позволяют получать промежуточные решения [19], другие используют ограничения на кардинальность [20], которые поддерживаются далеко не каждым SMT-решателем, а третьи эффективны только для небольших задач [21]. Первое обстоятельство исключает возможность получить потенциально хорошее, но неоптимальное решение, если оптимальное решение не было найдено за разумное время. Второе обстоятельство не позволяет использовать большинство известных SMT-решателей для решения задачи

MaxSMT, тогда как согласно соревнованиям SMT-COMP [24] на разных теориях наилучшие результаты показывают разные SMT-решатели. Третье обстоятельство делает алгоритмы непригодными для производственных задач.

В качестве SMT-решателей, на базе которых построен режим портфолио для задачи MaxSMT, были выбраны Z3 [6], Yices [7] и Bitwuzla [8]. Эти проекты давно разрабатываются, предлагают поддержку широкого набора теорий, а также демонстрируют хорошую производительность на соревнованиях SMT-COMP.

Режим портфолио реализован в рамках Kotlin-библиотеки KSMT [22], предоставляющей универсальный программный интерфейс для доступа к набору SMT-решателей. Эта библиотека позволяет реализовать алгоритм решения задачи MaxSMT один раз и использовать его с разными SMT-решателями. Кроме того, проект KSMT реализует конвертеры формул из конкретного представления SMT-решателя в универсальный формат библиотеки, что упрощает реализацию режима портфолио задачи MaxSMT.

В качестве базового алгоритма использован MaxSAT-алгоритм PrimalDualMaxRes. Он состоит из двух шагов: шаг получения промежуточного решения и шаг преобразования утверждений с учетом промежуточного решения с целью найти лучшие решения на следующих итерациях. Эти шаги повторяются до тех пор, пока не будет найдено оптимальное решение. Алгоритм PrimalDualMaxRes способен выдавать субоптимальные решения в условиях ограниченного времени. Также он отличается высокой скоростью сходимости. Целесообразность его использования в данном исследовании связана с возможностью замены вызовов SAT-решателей на вызовы SMT-решателей, сохранив все те преимущества, которые обозначены выше, и применив их для задачи MaxSMT.

3. Обзор решения

На рис. 1 представлена схема взаимодействия между разработанным MaxSMT-решателем и используемыми SMT-решателями. Формулы, специфичные для конкретного SMT-решателя, преобразуются в универсальный формат библиотеки KSMT, с которым работает MaxSMT-решатель. Решатель задачи MaxSMT позволяет устанавливать ограничения на время решения задачи, при этом полученные решения содержат информацию о том, являются ли они оптимальными. MaxSMT-решатель может быть сконфигурирован так, чтобы задача MaxSMT решалась с использованием любого из SMT-решателей, поддерживаемых KSMT. Решатель реализует адаптированную под решение задачи MaxSMT версию алгоритма PrimalDualMaxRes. MaxSMT-решатель может быть использован для работы в режиме портфолио.

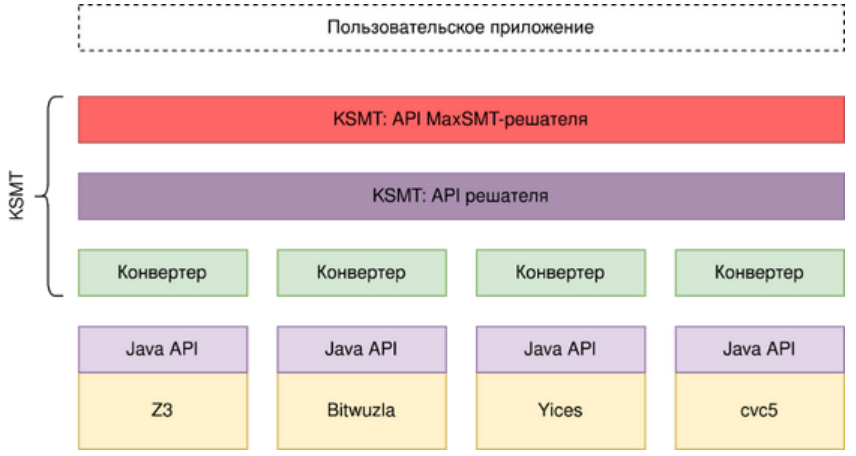

Рис. 1. Взаимодействие MaxSMT-решателя с используемыми SMT-решателями.
Fig. 1. Interaction of the MaxSMT solver with the used SMT solvers.

4. Адаптация алгоритма PrimalDualMaxRes

Поскольку процесс построения промежуточных решений в оригинальном алгоритме PrimalDualMaxRes ограничен работой с булевыми формулами, этот шаг заменен на семантически аналогичный, но позволяющий работать с формулами теорий первого порядка. Опишем, как это делается.

Пусть даны множества $H = \{ h_1, h_2, \dots, h_n \}$ и $S = \{ s_1, s_2, \dots, s_n \}$, которые включают в себя утверждения в контексте решения задачи MaxSMT – без весов и с весами соответственно. Пусть также имеется множество SLS для хранения решений, найденных к текущему моменту. На этом множестве задано отношение частичного порядка, определяемое суммой весов выполнимых утверждений. Изначально $SLS = \emptyset$. Далее проверяется выполнимость формулы $F_1 \leftarrow \bigwedge_{h \in H} h \wedge (s_1 \wedge s_2)$.

В том случае, если эта формула невыполнима, происходит переход ко второму и последнему шагу первоначального алгоритма PrimalDualMaxRes.

При условии выполнимости формулы получено промежуточное решение M. Если $M > \sup SLS$, то элемент M добавляется в множество SLS. Далее проверяется выполнимость формулы $F_2 \leftarrow F_1 \wedge (s_1 \wedge s_2 \wedge \dots \wedge s_{k+2})$. Поиск оптимального числа утверждений k, добавляемых на каждом шаге, требует дальнейшего исследования. Шаг адаптированного алгоритма повторяется до тех пор, пока не будет получено невыполнимое множество утверждений или не станет ясно, что все утверждения из множества S выполнимы.

В первом случае осуществляется переход к следующему шагу исходного алгоритма PrimalDualMaxRes. Во втором выполнение модифицированного алгоритма завершается с найденным оптимальным решением, соответствующим $\sup SLS$.

5. Реализация режима портфолио

Представленное решение расширяет реализацию режима портфолио решателей в KSMT, изначально созданную для задачи SMT, ориентируясь на решение задачи MaxSMT. Портфолио MaxSMT-решателей управляется запускающим его процессом. Каждый MaxSMT-решатель, используемый в портфолио, работает в отдельном процессе – см. рис. 2.

Обсудим детали реализации решения. Для управления портфолио SMT-решателей KSMT реализует класс PortfolioSolver. Этот класс обеспечивает выполнение набора стандартных операций (добавление утверждений, проверка их выполнимости и т.д.) для каждого решателя из портфолио в порядке, заданном пользователем. В KSMT были внесены изменения для того, чтобы добавленная операция решения MaxSMT не нарушала этот порядок (MaxSMT-решатель в KSMT наследует операции SMT-решателя). В класс PortfolioSolver также добавлена стратегия обработки решений: если один из решателей возвращает оптимальное решение, оно принимается в качестве ответа; в противном случае возвращается решение от того решателя, который нашел наилучшее решение за отведенное время.

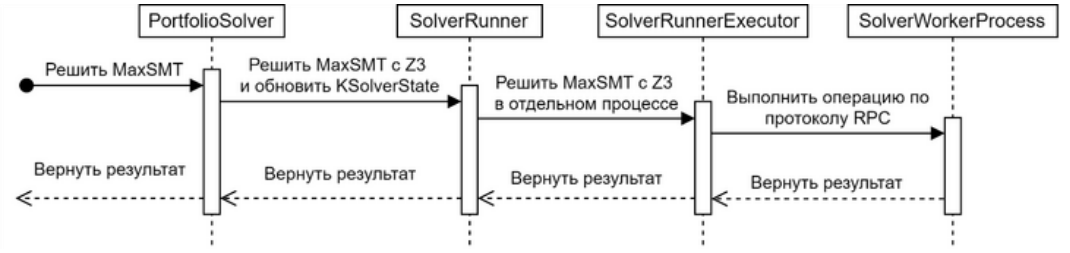

Рис. 2. Работа MaxSMT-решателя в режиме портфолио.
Fig. 2. MaxSMT solver operation in portfolio mode.

Объект класса SolverRunner хранит состояние решателя в процессе, который управляет портфолио. Это позволяет пересоздать процесс решателя и восстановить его состояние в случае зависания или неожиданного завершения. В библиотеку KSMT был также добавлен механизм восстановления утверждений с весами, чтобы обеспечить возможность восстановления состояния MaxSMT-решателя.

Объект класса SolverRunner перенаправляет запросы на выполнение операций экземпляру класса SolverExecutor, который затем направляет их по протоколу RPC в процесс решения задачи MaxSMT. В свою очередь, класс SolverWorkerProcess описывает действия, выполняемые в процессе решения задачи MaxSMT.

6. Эксперименты

Эксперименты осуществлялись на агентах сервиса GitHub [25] с процессором Intel Xeon E5, 2.6 GHz (4 ядра) и 16 Гб RAM, под управлением Linux (дистрибутив Ubuntu 22.04).

Поскольку в сообществе отсутствовал тестовый набор данных (benchmark) для задачи MaxSMT, то он был разработан в рамках нашего исследования. Это было сделано следующим образом. Сначала был взят инкрементальный тестовый набор данных 2023 года для задачи SMT [26]. Затем этот набор был преобразован для решения задачи MaxSMT путем назначения весов некоторым утверждениям. Веса назначались утверждениям, добавленным после выполнения первой операции push. Наконец, при помощи MaxSMT-решателя νZ [21] (часть проекта Z3), работавшего с ограничением времени в пять минут, были сгенерированы оптимальные решения задачи MaxSMT для примеров из преобразованного инкрементального тестового набора данных. Таким образом, получившийся тестовый набор включает примеры с бесквантворными фрагментами следующих теорий: массивов, битовых векторов, неинтерпретированных функций, чисел с плавающей точкой, линейной целочисленной и рациональной арифметик, а также различные комбинации этих теорий.

Эксперименты проводились для пяти следующих MaxSMT-решателей.

- Существующий MaxSMT-решатель νZ в стандартной конфигурации.
- Наш подход, работающий на каждом из SMT-решателей по отдельности – Z3, Yices и Bitwuzla.
- Наш подход, работающий в режиме портфолио с решателями Z3, Yices и Bitwuzla.

Представительность этого набора MaxSMT-решателей связана с тем, что входящие в него решатели основаны на производительных SMT-решателях и реализуют алгоритмы решения задачи MaxSMT с высокой скоростью сходимости.

В качестве метрик оценки качества решения было взято время работы, а также полученная оценка – значение из отрезка от нуля до единицы, вычисляемое как отношение суммы весов выполнимых утверждений, найденной решателем за отведенный промежуток времени, к наибольшей возможной сумме весов выполнимых утверждений. Было поставлено два эксперимента: в оптимальном и субоптимальном режимах. Целью первого эксперимента было оценить, насколько успешно представленный в работе MaxSMT-решатель справляется с набором примеров по сравнению с существующими решениями, а также проверить, может ли режим портфолио обеспечить заметный прирост производительности при работе с различными теориями. Целью второго эксперимента было убедиться, что если в условиях ограниченного времени MaxSMT-решатель не успевает найти оптимальное решение, то он может вернуть промежуточное, которое зачастую может оказаться близким к оптимальному.

В рамках каждого эксперимента было сделано по семь запусков. Все графики строились по средним значениям. Время работы всех решателей измерялось в том же процессе, в котором они запускались. Для решателя, работающего в режиме портфолио, время измерялось процессом, который этим портфолио управлял. Это позволило учесть затраты времени на взаимодействие между процессом-координатором и процессом решения задачи MaxSMT.

6.1 Оптимальный режим

В оптимальном режиме задача считается успешно решенной только в том случае, если найдено оптимальное решение.

В этом режиме на решение примера устанавливалось ограничение в 60 секунд. По результатам запусков построен график, который отражает зависимость количества примеров от времени, не дольше чем за которое каждый из них был решен – см. график, представленный на рис. 3. Стоит отметить, что для каждого примера было вычислено лучшее время решения, полученное в рамках запуска решателей. Именно эти значения использовались при построении функции, обозначенной на графике как «Лучшее».

Анализ графика с рис. 3 показывает, что решатель νZ смог решить около двухсот таких примеров, что на каждый из них он затратил менее пяти миллисекунд. В то же время у остальных решателей либо отсутствуют примеры, решенные менее чем за пять миллисекунд, либо их всего несколько. Для других решателей отметка в двести примеров достигается при значении времени в десять миллисекунд, тогда как νZ за это время решает уже триста примеров. Это можно объяснить тем, что исходная реализация MaxSMT-решателя проекта Z3 (νZ) не расходует ресурсы на преобразование формул в формат библиотеки KSMT, а также на взаимодействие с другими процессами, как это делает портфолио решателей.

Можно заметить, что начиная с какого-то момента, с ростом времени количество успешно решенных примеров у решателя, работающего в режиме портфолио, начинает увеличиваться быстрее, чем у остальных решателей. Это обусловлено тем, что накладные расходы на запуск портфолио оказываются незначительными по сравнению с выигрышем во времени, который он дает. График на рис. 3. демонстрирует, что функция, отражающая результаты измерений для портфолио решателей, наиболее близко соответствует функции, отражающей лучший результат.

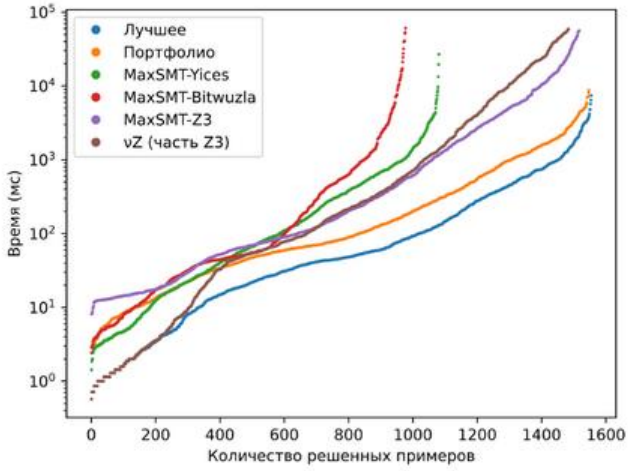

Рис. 3. Режим портфолио: среднее количество решенных примеров не дольше, чем за время (оптимальный режим).

Fig. 3. Portfolio mode: the average number of solved examples is no longer than in time (optimal mode).

В ходе проведенного эксперимента также было установлено количество успешно решенных примеров для каждого из MaxSMT-решателей. Кроме этого, было рассчитано среднее время, затраченное на решение каждого примера (см. табл. 1).

Табл. 1 демонстрирует, что решатель в режиме портфолио превосходит другие MaxSMT-решатели по количеству успешно решенных примеров. Меньше всего примеров решено

MaxSMT-решателем, использующим SMT-решатель Bitwuzla. Это связано с тем, что Bitwuzla, как и Yices, поддерживает не все теории из тестового набора данных.

Согласно данным из табл. 1, лучшее среднее время работы демонстрирует решатель на базе Yices. Портфолио MaxSMT-решателей несколько уступает по этому показателю. Однако решатель в режиме портфолио успешно справляется с гораздо бóльшим набором задач. Он также успешно справляется с бóльшим числом примеров, чем νZ, при этом в восемь раз превосходя его по среднему времени решения примера. В то же время среднее время работы νZ заметно превышает время работы MaxSMT-решателя, использующего решатель Z3 библиотеки KSMT. Это может быть обусловлено наличием неоднозначных операций над unsat-ядрами [27], которые выполняются в рамках нативной реализации Z3. Эти операции позволяют получить выигрыш в производительности на каких-то примерах, но в то же время на других примерах ведут к деградации.

Табл. 1. Среднее количество решенных примеров (оптимальный режим).
Table 1. Average number of solved examples (optimal mode).

Решатели	Кол-во решенных примеров	Среднее время на пример (мс)
Портфолио решателей	1546 ± 2	495 ± 11
Yices	1080 ± 2	459 ± 24
Bitwuzla	943 ± 46	883 ± 139
Z3	1508 ± 4	2582 ± 57
νZ (часть Z3)	1479 ± 3	3987 ± 40

6.2 Субоптимальный режим

В субоптимальном режиме наш подход учитывает любое решение, даже если оно не является оптимальным. При этом в наших экспериментах на решение примера выделялось 10 секунд.

В результате запусков построен график, представленный на рис. 4, который иллюстрирует минимальное количество примеров, на которых полученная оценка не меньше соответствующего значения. На графике отражены показатели для различных MaxSMT-решателей, а также лучший результат по каждому из примеров. Все решатели смогли набрать на девятистах примерах максимальные оценки, и поэтому эти примеры не отражены на графике. График показывает, что решатель в режиме портфолио справился с наибольшим количеством примеров, набрав максимальные оценки. И только на нескольких примерах он получил оценки, отличные от единицы, однако близкие к ней. Этот показатель близок к лучшему из возможных. MaxSMT-решатель, использующий SMT-решатель Yices, тоже практически на всех примерах набрал оценки, равные единице. Однако, поскольку Yices не поддерживает некоторые теории, он запустился не на всех примерах. Остальные решатели на многих примерах получают оценки, отличные от единицы, причем среди этих значений немало таких, которые значительно отличаются от единицы.

В ходе проведенного эксперимента также была вычислена сумма оценок, набранных решателем на тестовом наборе данных, и среднее время решения каждого примера (табл. 2).

Табл. 2. Средние сумма оценок и время, затраченное на пример (субоптимальный режим).
Table 2. Average scores and time spent on the example (suboptimal mode).

Решатели	Оценки	Среднее время на пример (мс)
Портфолио решателей	1555 ± 1	522 ± 9
Yices	1088 ± 1	426 ± 10
Bitwuzla	979 ± 3	727 ± 32
Z3	1505 ± 3	1916 ± 27
νZ (часть Z3)	1473 ± 3	2451 ± 12

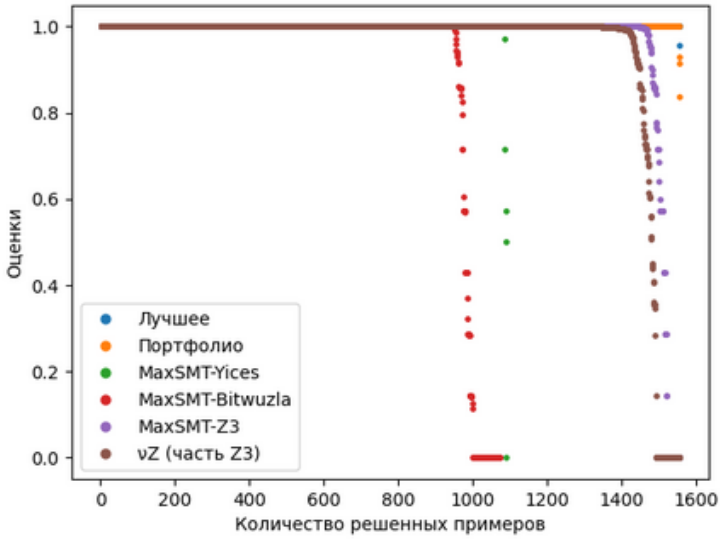

Рис. 4. Распределение оценок в зависимости от количества примеров (субоптимальный режим)
Fig. 4. Distribution of the score depending on the number of examples (suboptimal mode)

Лучшее среднее время работы демонстрирует решатель на базе Yices. Тем не менее, решатель в режиме портфолио, получает лучшие оценки на тестовом наборе данных, хоть и при несколько большем среднем времени работы, чем Yices. Портфолио решателей справляется с примерами в четыре раза быстрее, чем νZ, и при этом набирает лучшие оценки на тестовом наборе данных.

7. Обзор существующих работ

К настоящему времени в сообществе не представлено ни одного MaxSMT-решателя, который бы поддерживал режим портфолио. Более того, в современных решателях реализация решения задачи MaxSMT тесно связана с конкретным SMT-решателем. Это лишает возможности выбрать наиболее подходящий для конкретной задачи SMT-решатель, что заметно снижает эффективность, а значит и применимость MaxSMT-решателей.

Одним из способов преодоления этого ограничения является использование универсальных интерфейсов (так называемых «оберток») для взаимодействия с различными SMT-решателями. К таким инструментам относятся metaSMT [28], JavaSMT [29], pySMT [30]. Они позволяют абстрагироваться от особенностей реализации конкретных решателей, что, в теории, позволяет реализовать алгоритм MaxSMT на более высоком уровне и использовать в процессе решения задачи любой поддерживаемый «оберткой» SMT-решатель. Однако на практике существующие инструменты пока не используют эту возможность.

Наиболее эффективным существующим MaxSMT-решателем является νZ. Этот решатель может выдавать не только оптимальные решения, но и промежуточные. Одним из ключевых преимуществ νZ является способность работать с большим набором теорий, включая такие популярные теории, как битовые вектора, массивы, линейная целочисленная и рациональная арифметики, неинтерпретированные функции, а также числа с плавающей точкой. Тем не менее, νZ ограничен использованием SMT-решателя Z3, что сужает его потенциальную применимость в случаях, где использование других SMT-решателей могло бы позволить показать лучшие результаты.

Стоит отметить OMT-решатель OptiMathSAT [31], основанный на SMT-решателе MathSAT [32] с закрытым исходным кодом. OptiMathSAT представляет собой инструмент для решения задачи OMT (Optimization Modulo Theories), которая заключается в поиске оптимального решения SMT-проблемы относительно целевых функций, описывающих ключевые показатели системы. Задача MaxSMT, для которой нетрудно определить целевую функцию, может быть сведена к задаче OMT [31]. Однако такой способ не является эффективным способом решения задачи MaxSMT, поскольку оптимизационный механизм OMT-решателя может выполнять избыточный перебор различных комбинаций, приводящих к одинаковым результатам [33].

8. Заключение

В данной работе представлена адаптированная версия алгоритма PrimalDualMaxRes для решения задачи MaxSAT, которая позволяет решать более общую задачу – MaxSMT. Модифицированный алгоритм реализован с использованием Kotlin-библиотеки KSMT, поддерживая возможность работы в режиме портфолио.

Эксперименты продемонстрировали, что использование режима портфолио позволяет решать большинство примеров быстрее. Также установлено, что на тестовом наборе данных портфолио MaxSMT-решателей превосходит по времени решения существующий MaxSMT-решатель *νZ* (проект Z3) не менее чем в четыре раза, демонстрируя мощность предложенного решения. В качестве развития данного исследования можно отметить возможность интеграции итогового MaxSMT-решателя для генерации тестов в специализированные среды разработки – встроенных систем реального времени [34], систем телевещания [35] и др.

Список литературы / References

[1]. Alouneh, S., Abed, S. E., Al Shayeji, M. H., Mesleh, R. A comprehensive study and analysis on SAT-solvers: advances, usages and achievements. Artificial Intelligence Review, vol. 52, 2019, pp. 2575–2601.

[2]. Misonizhnik, A., Mordvinov, D. On Satisfiability of Nominal Subtyping with Variance // Proceedings of 33rd European Conference on Object-Oriented Programming (ECOOP 2019), 2019.

[3]. Barrett, C., Sebastiani, R., Seshia, S. A., Tinelli, C. Satisfiability modulo theories // Handbook of satisfiability. IOS Press, 2021, pp. 1267–1329.

[4]. Костюков, Ю. О., Батоев, К. А., Мордвинов, Д. А., Костицын, М. П., Мисонижник, А. В. Автоматическое доказательство корректности программ с динамической памятью. Труды ИСП РАН, том 31, вып. 5, 2019 г., стр. 37–62.

[5]. Barrett, C., Tinelli, C., Barbosa, H., Niemetz, A., Preiner, M., Reynolds, A., Zohar, Y. Satisfiability Modulo Theories: A Beginner's Tutorial. International Symposium on Formal Methods, 2024, pp. 571– 596.

[6]. De Moura, L., Bjørner, N. Z3: An efficient SMT solver. In International conference on Tools and Algorithms for the Construction and Analysis of Systems, 2008, March, pp. 337–340.

[7]. Dutertre, B. Yices 2.2. In International Conference on Computer Aided Verification, 2014, pp. 737–744.

[8]. Niemetz, A., Preiner, M. Bitwuzla. In International Conference on Computer Aided Verification, 2023, pp. 3–17.

[9]. Pasareanu, C. S., Phan, Q. S., Malacaria, P. Multi-run side-channel analysis using Symbolic Execution and Max-SMT // Proceedings of IEEE 29th Computer Security Foundations Symposium (CSF). 2016, pp. 387–400.

[10]. Brockschmidt, M., Larra, D., Oliveras, A., Rodrıguez-Carbonell, E., Rubio, A. Compositional safety verification with Max-SMT // Formal Methods in Computer-Aided Design (FMCAD). 2015, pp. 33–40.

[11]. Terra-Neves, M., Machado, N., Lynce, I., Manquinho, V. Concurrency debugging with MaxSMT // Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, No. 01, 2019, pp. 1608–1616.

[12]. Larraz, D., Nimkar, K., Oliveras, A., Rodríguez-Carbonell, E., Rubio, A. Proving non-termination using Max-SMT // Proceedings of 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, pp. 779–796.

[13]. Larraz, D., Oliveras, A., Rodríguez-Carbonell, E., Rubio, A. Proving termination of imperative programs using Max-SMT. In 2013 Formal Methods in Computer-Aided Design, 2013, pp. 218–225.
[14]. Albert, E., Gordillo, P., Rubio, A., Schett, M. A. Synthesis of super-optimized smart contracts using max-smt. In International conference on computer aided verification, 2020, pp. 177–200.
[15]. Li, C. M., Manya, F. MaxSAT, hard and soft constraints. In Handbook of satisfiability. IOS Press, 2021, pp. 903–927.
[16]. Мисонижник А.В., Бабушкин А.А., Морозов С.А., Костюков Ю.О., Мордвинов Д.А., Кознов Д.В. Автоматическое тестирование LLVM-программ со сложными входными структурами данных. Труды ИСП РАН, том 34, вып. 4, 2022 г., стр. 49–62.
[17]. Chen, T., Zhang, X. S., Guo, S. Z., Li, H. Y., Wu, Y. State of the art: Dynamic symbolic execution for automated test generation. Future Generation Computer Systems, vol. 29(7), 2013, pp. 1758–1773.
[18]. Bjorner, N., Narodytska, N. Maximum satisfiability using cores and correction sets // Proceedings of Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015, pp. 246–252.
[19]. Narodytska, N., Bacchus, F. Maximum satisfiability using core-guided MaxSAT resolution // Proceedings of the AAAI Conference on Artificial Intelligence, vol. 28, No. 1, 2014.
[20]. Morgado, A., Dodaro, C., Marques-Silva, J. Core-Guided MaxSAT with Soft Cardinality Constraints. Principles and Practice of Constraint Programming, 2014, pp. 564–573.
[21]. Bjørner, N., Phan, A. D., Fleckenstein, L. 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, 2015, pp. 194–199.
[22]. KSMT, https://github.com/UnitTestBot/ksmt (дата обращения: 01.02.2025).
[23]. MaxSMT benchmark, https://github.com/viktoriia-fomina/ksmt/releases/tag/0.0.0 (дата обращения: 01.02.2025).
[24]. SMT-COMP 2023, https://smt-comp.github.io/2023/ (дата обращения: 01.02.2025).
[25]. GitHub hosted runners, https://docs.github.com/en/actions/using-github-hosted-runners/about-github-hosted-runners/about-github-hosted-runners (дата обращения: 01.02.2025).
[26]. Тестовый набор данных для задачи SMT, https://smt-lib.org/benchmarks.shtml (дата обращения: 01.02.2025).
[27]. Guthmann, O., Strichman, O., Trostanetski, A. Minimal unsatisfiable core extraction for SMT. In 2016 Formal Methods in Computer-Aided Design (FMCAD), 2016, pp. 57–64.
[28]. Haedicke, F., Frehse, S., Fey, G., Große, D., Drechsler, R. metaSMT: Focus on Your Application not on Solver Integration. 2011, p. 47.
[29]. Baier, D., Beyer, D., Friedberger, K. JavaSMT 3: Interacting with SMT solvers in Java // Proceedings of International Conference on Computer Aided Verification, 2021, pp. 195-208. DOI: 10.1007/978-3-030-81688-9_9.
[30]. Gario, M., Micheli, A. PySMT: a solver-agnostic library for fast prototyping of SMT-based algorithms // Proceedings of SMT workshop, 2015.
[31]. Sebastiani, R., Trentin, P. OptiMathSAT: A tool for optimization modulo theories. Journal of Automated Reasoning, 64(3), 2020, pp. 423–460.
[32]. Cimatti, A., Griggio, A., Schaafsma, B. J., Sebastiani, R. The mathsat5 smt solver. In International Conference on Tools and Algorithms for the Construction and Analysis of Systems, 2013, pp. 93–107.
[33]. Sebastiani, R., Trentin, P. On optimization modulo theories, MaxSMT and sorting networks // Proceedings of International Conference on Tools and Algorithms for the Construction and Analysis of Systems, 2017, pp. 231–248.
[34]. Terekhov A.N., Romanovskii K.Y., Koznov D.V., Dolgov P.S., Ivanov A.N. RTST++: Methodology and a CASE tool for the development of information systems and software for real-time systems. Programming and Computer Software, 1999, 25 (5), pp. 276–281.
[35]. Кознов Д.В., Перегудов А.Ф., Бугайченко Д.Ю., Чернятчик Р.И., Казакова А.С., Павлинов А.А. Визуальная среда проектирования систем телевизионного вещания. Системное программирование, 2006, 2(1), стр. 142–168.

Информация об авторах / Information about authors

Виктория Викторовна ФОМИНА – аспирант математико-механического факультета СПбГУ. Сфера научных интересов: анализ программ.

Viktoriia Viktorovna FOMINA is a postgraduate student of the Faculty of Mathematics and Mechanics, St. Petersburg State University. Her research interests include program analysis.

Валентин СОБОЛЬ – выпускник Санкт-Петербургского Политехнического университета им. Петра Великого, преподаватель университета ИТМО. Научные интересы: статический анализ программ, символьное исполнение.

Valentyn SOBOL is a lecturer at ITMO University and a graduate of Peter the Great St. Petersburg Polytechnic University. His scientific interests include static analysis of programs and symbolic execution.

Дмитрий Владимирович КОЗНОВ – доктор технических наук, профессор кафедры системного программирования Санкт-Петербургского государственного университета, Сфера научных интересов: программная инженерия, модельно-ориентированная разработка программного обеспечения, программные данные, машинное обучение.

Dmitry Vladimirovich KOZNOV – Dr. Sci. (Tech.), Associate Professor, Professor St. Petersburg State University (SPbSU). Research interests: software engineering, model-driven software development, program data, machine learning.