Использование метода опорных векторов для оценки полезности конфликтных дизъюнктов в CDCL-выводе

О. С. Заикин, С. Е. Кочемазов

Аннотация—**Многие современные алгоритмы для решения проблемы булевой выполнимости (SAT) основаны на алгоритме CDCL. В процессе своей работы они генерируют большое количество т. н. конфликтных дизъюнктов, соответствующих пройденным ветвям дерева возможных решений. Часть конфликтных дизъюнктов необходимо время от времени удалять, чтобы поддерживать высокую скорость работы алгоритма. Таким образом, возникает задача оценки полезности конфликтных дизъюнктов, чтобы выявить какие из них оставлять, а какие нет. В настоящем исследовании для решения этой задачи предложена эвристика, основанная на использовании метода опорных векторов. На первом этапе формируется семейство упрощенных версий исходной SAT-задачи, затем они решаются с помощью SAT-решателя. На втором этапе осуществляется обучение машины опорных векторов, при этом конфликтный дизъюнкт считается полезным, если он не был удален к моменту нахождения решения хотя бы одной задачи из семейства. На третьем этапе при решении исходной SAT-задачи полезность некоторых дизъюнктов оценивается с помощью обученной машины опорных векторов. Базируясь на предложенной эвристике, был реализован модифицированный вариант одного из современных SAT-решателей, основанного на CDCL. Согласно проведенным вычислительным экспериментам, модифицированная версия решателя работает эффективнее оригинала на нескольких семействах сложных экземпляров SAT.**

Ключевые слова—проблема булевой выполнимости, CDCL, метод опорных векторов.

I. Введение

В данной статье мы рассматриваем проблему булевой выполнимости (SAT) в её поисковом варианте: по произвольной булевой формуле необходимо найти выполняющий ее набор, либо доказать, что такой набор не существует [1]. Несмотря на NP-трудность этой задачи, в последние 30 лет наблюдается существенный прогресс в разработке алгоритмов ее решения. Благодаря этому прогрессу, большое количество задач из различных областей, таких как биоинформатика, криптоанализ, программная аппаратная верификация, были успешно сведены к SAT и решены с помощью соответствующих алгоритмов.

Произвольный экземпляр проблемы булевой выполнимости называется SAT-задачей. SAT-подход к решению некоторой задачи состоит в сведении этой задачи к SAT с последующим ее решением с помощью специализированных программ, называемых SAT-решателями. При этом сведение к SAT обычно строится таким образом, что от найденного решения SAT-задачи можно эффективно получить решение исходной задачи.

На сегодняшний день, наиболее эффективным базовым алгоритмом решения SAT является CDCL (Conflict Driven Clause Learning [2]), который был предложен в 1996 году. CDCL реализует обход двоичного дерева решений, которое формируется исходя из ограничений исходной булевой формулы. Во время своей работы алгоритм CDCL генерирует новые ограничения, которые могут существенно ускорить процесс решения. При этом на практике часть таких дополнительных ограничений нужно периодически удалять, во-первых, чтобы ограничить количество используемой памяти, а во-вторых, для поддержания высокого темпа обхода дерева решений.

В современных SAT-решателях, основанных на CDCL, многие процедуры базового алгоритма работают под управлением разнообразных эвристик. Их разработка является актуальным направлением исследований, даже если в результате получается добиться ускорения лишь на некоторых семействах SAT-задач. Некоторые из этих эвристик основаны на алгоритмах машинного обучения, которые в свою очередь также интенсивно развиваются последние десятилетия. Среди них можно особо отметить эвристику LRB (Learning Rate based Branching [3]), согласно которой выбор направления обхода двоичного дерева решений сводится к задаче о многоруком бандите [4] и решается при помощи алгоритма машинного обучения с подкреплением.

В настоящем исследовании предлагается эвристика, которая ориентирована на оценку полезности дополнительных ограничений, генерируемых в процессе работы CDCL. Эта эвристика основана на использовании метода опорных векторов [5], который относится к машинному обучению с учителем. Предлагается модификация современного SAT-решателя Glucose, основанная на разработанной эвристике. При этом Glucose запускается как внешний исполняемый файл из основной программы. Модифицированная версия решателя сравнивается с

Статья получена 28 сентября 2019. Исследование поддержано советом по грантам Президента Российской федерации (грант МК-4155.2018.9, стипендия СП-2017.2019.5) и Российским фондом фундаментальных исследований (грант 19-07-00746-а).

Олег Сергеевич Заикин, старший научный сотрудник Института динамики систем и теории управления им. В.М. Матросова СО РАН (e-mail: zaikin.icc@gmail.com).

Степан Евгеньевич Кочемазов, научный сотрудник Института динамики систем и теории управления им. В.М. Матросова СО РАН (e-mail: veinamond@gmail.com).

оригиналом на двух семействах трудных SAT-задач. В основном эти семейства состоят из задач криптоанализа.

II. Алгоритм CDCL

В данном разделе описаны некоторые свойства алгоритма CDLC, достаточные для понимания подхода, предлагаемого в настоящей работе.

На практике при решении SAT-задач булева формула обычно представляется в виде конъюнктивной нормальной формы (КНФ). Напомним, что литералом называется булева переменная либо её отрицание. Элементарный дизъюнкт – это дизъюнкция литералов. КНФ – это конъюнкция элементарных дизъюнктов. Каждый такой дизъюнкт можно рассматривать как отдельное ограничение, которое необходимо удовлетворить для того, чтобы найти набор значений переменных, выполняющий данную КНФ. Схематично, получив на вход некоторую КНФ, алгоритм CDCL работает следующим образом.

1. Выбирается неозначенная булева переменная, значение которой будет угадано. Конкретное значение определяется эвристически. Если все переменные означены, то КНФ выполнима и текущие значения переменных формируют её выполняющий набор.
2. После угадывания переменной производится т.н. распространение булевых ограничений: значение угаданной переменной подставляется во все дизъюнкты, в которые она входит. Часто это приводит к выводу значений других неозначенных переменных, которые подставляются в соответствующие дизъюнкты, и так далее. Данный шаг заканчивается одной из двух альтернатив – если в рамках шага 2 для некоторой переменной были выведены противоположные значения, то такая ситуация помечается как конфликт и обрабатывается на шаге 3. Иначе происходит возврат на шаг 1 и выбирается следующая переменная для угадывания.
3. При помощи специальных процедур осуществляется анализ причин конфликта, результатом которых является т. н. конфликтный дизъюнкт, который, будучи приписанным к дизъюнктам исходной КНФ, запрещает возникновение аналогичной конфликтной ситуации. После приписывания конфликтного дизъюнкта к базе дизъюнктов осуществляется отмена означивания некоторого числа угаданных переменных. В случае если результатом анализа конфликтов является пустой дизъюнкт, то КНФ является невыполнимой.

На практике база конфликтных дизъюнктов может вырастать до огромного размера, поэтому периодически некоторые из них необходимо удалять. При этом очень важной проблемой является выбор конфликтных дизъюнктов для удаления. Таким образом, возникает задача оценки полезности конфликтных дизъюнктов, которую решают при помощи различных эвристик.

III. Эвристика оценки полезности конфликтных дизъюнктов при помощи метода опорных векторов

Метод опорных векторов [5] – это класс алгоритмов машинного обучения с учителем, использующихся для решения задач классификации и регрессионного анализа. В случае бинарной классификации идея метода состоит в том, что в процессе обучения каждый элемент обучающей выборки представляется как вектор в p-мерном пространстве. Затем строится гиперплоскость размерности $p-1$ таким образом, чтобы максимально разделить элементы из двух классов. Если это удается сделать так, что все элементы первого класса оказываются по одну сторону от гиперплоскости, а все элементы второго класса по другую сторону, то это случай линейной разделимости. Если такую гиперплоскость построить невозможно, то это случай линейной неразделимости. При этом и в случае линейной неразделимости метод опорных векторов может дать хорошую точность решения некоторых задач классификации. После построения гиперплоскости данный на вход элемент, который нужно классифицировать, помещается (в виде вектора) в p-мерное пространство, а его класс назначается в соответствии с тем, по какую сторону от разделяющей гиперплоскости он оказался.

Для решения упомянутой в предыдущем разделе задачи оценки полезности конфликтных дизъюнктов предлагается эвристика, основанная на методе опорных векторов. Пусть дана некоторая КНФ над множеством из N булевых переменных. Также пусть дана обучающая выборка, которая содержит конфликтные дизъюнкты для данной КНФ, разделенные на два класса: полезные и бесполезные. На этапе обучения все конфликтные дизъюнкты представляются как булевы векторы длины N и помещаются в пространство, которое является N-мерным булевым гиперкубом. Затем с помощью метода опорных векторов строится гиперплоскость размерности N-1, нацеленная на разделение полезных конфликтных дизъюнктов и бесполезных. В общем случае при этом нельзя гарантировать линейную разделимость. После окончания обучения полезность нового конфликтного дизъюнкта (из тестовой выборки) оценивается путем его подстановки в пространство, заполненное на этапе обучения. При этом класс этому конфликтному дизъюнкту назначается в зависимости от того, по какую сторону от гиперплоскости он оказался.

На практике основная сложность применения предложенной эвристики заключается в том, что обычно крайне сложно сформировать обучающую выборку, где конфликтные дизъюнкты разделены на полезные и бесполезные по некоторому содержательному критерию. В рамках данного исследования нами был сформулирован следующий подход.

Пусть даны исходная КНФ, SAT-решатель, а также заданы значения параметров k, t_1, t_2, s, d. Алгоритм состоит из следующих шагов.

1. Формируется семейство упрощенных SAT-задач, полученных с помощью подстановки в исходную КНФ случайных значений k, k<N случайно выбранных переменных.
2. Все SAT-задачи, сформированные на шаге 1, решаются с помощью SAT-решателя с лимитом

времени t_1.

3. Для каждой SAT-задачи, которая была решена на шаге 2 до достижения лимита времени t_1, анализируется история генерации и удаления конфликтных дизъюнктов в процессе работы SAT-решателя. Конфликтный дизъюнкт считается бесполезным, если он был удален к моменту нахождения решения подзадачи из семейства. В противном случае он считается полезным. Таким образом формируется обучающая выборка.
4. С помощью описанной выше эвристики осуществляется обучение на сформированной обучающей выборке с использованием метода опорных векторов.
5. На исходной КНФ запускается SAT-решатель на время t_2, при этом все сгенерированные конфликтные дизъюнкты сохраняются.
6. С помощью предложенной эвристики оценивается полезность всех конфликтных дизъюнктов, сохраненных на шаге 5. К исходной КНФ добавляется d процентов дизъюнктов, которые были классифицированы как полезные, и размер которых при этом не превышает значения s.
7. На полученной КНФ запускается SAT-решатель.

Прокомментируем предложенный алгоритм. На шаге 3 мы полагаемся на эффективность современных CDCL-решателей, которые к моменту нахождения решения подзадачи стремятся удалить все лишние конфликтные дизъюнкты и оставить только полезные. На шаге 6 используется идея, предложенная в [6], согласно которой приписывание к исходной КНФ некоторых конфликтных дизъюнктов в качестве основных ограничений может для некоторых семейств SAT-задач привести к уменьшению времени решения.

IV. Программная реализация и вычислительные эксперименты

Предложенный в предыдущем разделе алгоритм был реализован в виде программы на языке C++. В качестве SAT-решателя был выбран Glucose [7], который многократно брал призовые места на специализированных соревнованиях. Конкретнее, была использована версия 4.1 данного решателя. Напомним, что изменения в исходный код SAT-решателя не вносились. Glucose запускался из базовой программы в качестве внешнего исполняемого файла, на вход которому подавались файлы с соответствующими КНФ. На шаге 3 история генерации и удаления конфликтных дизъюнктов выводилась в файл в формате DRAT [8] с помощью функционала, имеющегося в Glucose.

Опишем выбранные значения параметров алгоритма. На шаге 1 рассматривается 61 вариант значений параметра k – от 20 до 80 включительно. Это было сделано из-за того, что для разных SAT-задач эффект от подстановки значений одного и того же количества переменных может привести к разным эффектам – в одном случае исходная SAT-задача может значительно упроститься, в другом – остаться практически такой же сложной для SAT-решателя. Для каждого значения k были выбраны случайным образом три множества из k переменных исходной КНФ, затем для каждого из таких множеств был случайным образом выбран один набор значений выбранных переменных. С помощью подстановки этих значений в исходную КНФ формируются 183 упрощенные SAT-задачи.

Значение лимита времени t_1 на решение ослабленных подзадач на шаге 2 было равно 20 секундам. Промежуток времени t_2, на который запускался SAT-решатель на шаге 5, был равен 1 минуте. На шаге 6 значения d и s были равны 10 и 15 соответственно, т.е. к исходной КНФ добавлялись 10 процентов дизъюнктов, которые были определены как полезные и размер которых при этом был не более 15.

При реализации шага 4 был использован класс *svm_c_linear_trainer* библиотеки Dlib [9], при этом у основных параметров lambda и C этого класса значения были 0.001 и 1000 соответственно.

Фактически, разработанная программа является SAT-решателем, базирующемся на Glucose. Чтобы сравнить эффективность работы модифицированного решателя с оригиналом, были сформированы два семейства КНФ. Первое из них состоит из подмножества КНФ, которые использовались для сравнения SAT-решателей на соревновании SAT Competition 2017. Конкретнее, были выбраны 20 КНФ, кодирующих задачу сборки кубика Рубика, а также 10 КНФ, кодирующих ослабленные задачи криптоанализа блочного шифра DES. В последних 10 КНФ ослабление было сделано путем подстановки правильных значений некоторых из 56 переменных, кодирующих секретный ключ шифра. Подставлялись значения от 16 до 34 переменных (с шагом 2).

Второе семейство состоит из КНФ, кодирующих задачи криптоанализа некоторых генераторов ключевого потока. Были взяты КНФ, предложенные ранее в статьях [6] и [10]. Всего в семействе по 10 КНФ для каждого из следующих генераторов ключевого потока:

1. генератор переменного шага (72-битный секретный ключ);
2. генератор переменного шага (96-битный секретный ключ);
3. модифицированный генератор переменного шага MASG (72-битный секретный ключ);
4. модифицированный генератор переменного шага MASG0 (72-битный секретный ключ);
5. суммирующий генератор (64-битный секретный ключ);
6. пороговый генератор (72-битный секретный ключи);
7. генератор Гиффорда (64-битный секретный ключ).

Итого второе семейство состоит из 70 КНФ. Как и в ряде предыдущих статей, например [11, 12], эти задачи криптоанализа были сведены к SAT с помощью программного комплекса Transalg [13].

Вычислительные эксперименты проводились на одном узле вычислительного кластера “Академик В.М. Матросов” СКЦ СО РАН [14]. На каждой из КНФ обоих семейств решатели были запущены с лимитом времени 1 сутки (86400 секунд) на одном ядре процессора.

На Рис. 1 и Рис. 2 показаны результаты для первого семейства. На Рис. 1 лимит времени установлен равным 5000 секундам, как это обычно принято на

соревнованиях SAT-решателей. Это было сделано для того, чтобы сравнить решатели на решении относительно простых SAT-задач. На Рис. 2 лимит времени равен 1 суткам. По оси *Y* указано время решения в секундах, по оси *X* указано количество решенных SAT-задач, при этом решенные SAT-задачи упорядочены по возрастанию времени решения. Чем правее и ниже график SAT-решателя, тем он лучше. Оригинальный SAT-решатель (Glucose-4.1) сравнивается с модифицированным вариантом (Glucose-4.1-mod).

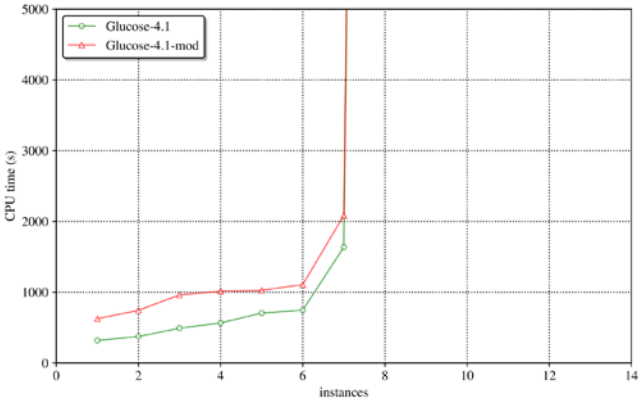

Рисунок 1. Результаты для первого семейства, лимит времени 5000 секунд.

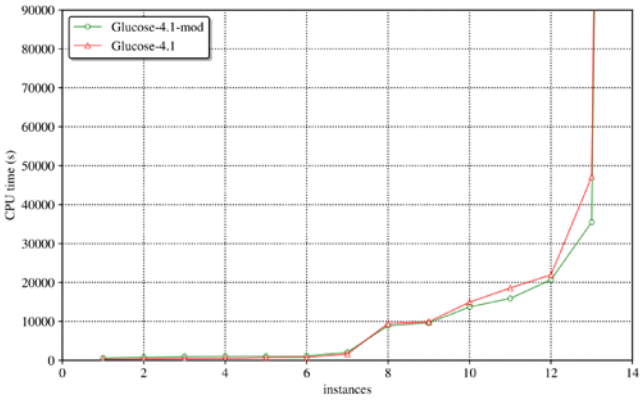

Рисунок 2. Результаты для первого семейства, лимит времени 1 день.

Исходя из результатов, на простых SAT-задачах первого семейства модифицированный вариант решателя показал себя хуже оригинала. Отметим, что на практике оказалось, что шаги 1-6 алгоритма обычно занимают несколько минут. Время работы этих предварительных шагов алгоритма также включено в общее время работы модифицированного решателя. При этом с лимитом 1 сутки модифицированный решатель обошел оригинал. Количество решенных SAT-задач оказалось одинаково, но модифицированная версия показала более низкое значение PAR-2, по которому принято сравнивать SAT-решатели. Значение PAR-2 для решателя на семействе SAT-задач с лимитом времени *t* равно сумме времени работы на решенных задачах плюс количество нерешенных задач, умноженное 2*t*. На первом семействе у Glucose-4.1 значение PAR-2 равно 1585935, а у Glucose-4.1-mod равно 1571623.

На Рис. 3 и 4 показаны результаты для второго семейства. На нем наблюдается та же картина, что и на первом семействе. На простых SAT-задачах исходный решатель оказался лучше, а на сложных SAT-задачах лучше себя показал модифицированный вариант. В частности, модифицированный вариант решил на 1 сложную задачу больше. На втором семействе у Glucose-4.1 значение PAR-2 равно 2471572, а у Glucose-4.1-mod равно 2466053. Таким образом, можно сделать вывод, что модифицированный алгоритм лучше оригинала ведет себя на некоторых классах сложных примеров SAT. Этим можно воспользоваться для решения практических задач, которые эффективно сводятся к SAT.

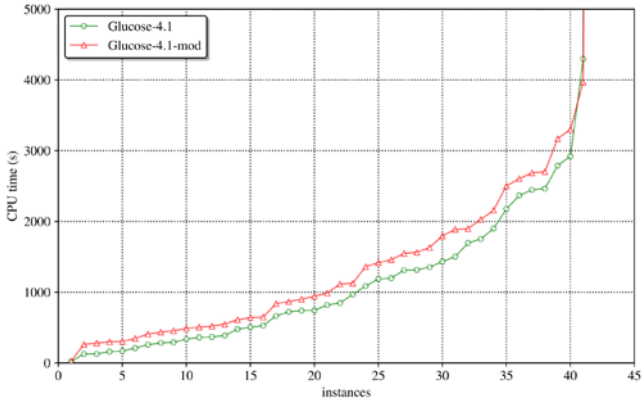

Рисунок 3. Результаты для второго семейства, лимит времени 5000 секунд.

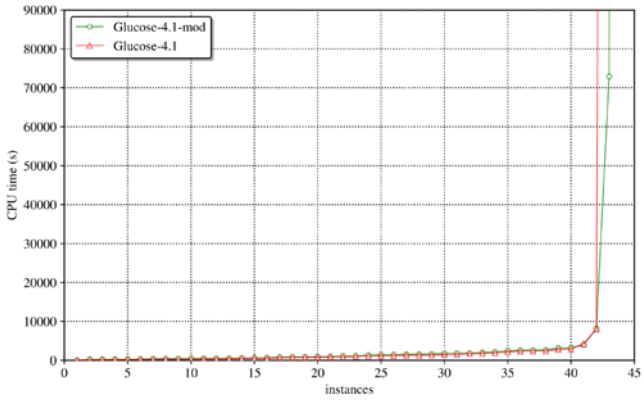

Рисунок 4. Результаты для второго семейства, лимит времени 1 день.

V. Заключение

В данной статье предложена новая эвристика оценки полезности конфликтных дизъюнктов, генерируемых в процессе работы алгоритма CDCL. На основе данной эвристики была разработана модификация SAT-решателя Glucose. Оказалось, что на простых SAT-задачах модификация не дает значительного прироста эффективности решения, в то время как на сложных SAT-задачах такой прирост есть.

В будущем планируется реализация предложенной эвристики непосредственно в CDCL-решателях для оценки полезности конфликтных дизъюнктов при чистках соответствующей базы.

Библиография

[1] M. R. Garey, D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co. New York, NY, USA, 1979.

[2] J. P. Marques-Silva, K. A. Sakallah, “GRASP-a new search algorithm for satisfiability,” in *Proc. IEEE International Conference on Computer-Aided Design (ICCAD)*, 1996, pp. 220–227.
[3] J. H. Liang, V. Ganesh, P. Poupart, and K. Czarnecki, “Learning rate based branching heuristic for SAT solvers,” in *Proc. Theory and Applications of Satisfiability Testing (SAT)*, Lecture Notes in Computer Science, vol. 9710, 2016, pp 123-140.
[4] M. N. Katehakis, A. F. Veinott, “The Multi-Armed Bandit Problem: Decomposition and Computation,” *Mathematics of Operations Research*, vol. 12 (2), 1995, pp. 262–268.
[5] C. Corinna, V. N. Vapnik, “Support-vector networks,” *Machine Learning*, vol. 20 (3), 1995, pp. 273–297.
[6] O. S. Zaikin, “A parallel SAT solving algorithm based on improved handling of conflict clauses,” In *Proc. International young scientists conference in high performance computing and simulation*, Procedia computer science, vol. 12083, 2017, pp. 103-111.
[7] G. Audemard, L. Simon, “On the Glucose SAT solver,” *International Journal on Artificial Intelligence Tools*, vol. 27 (1), 2018, pp. 1-25.
[8] M. Heule, W. A. Hunt Jr., and N. Wetzler, “Trimming while checking clausal proofs,” In *Proc. FMCAD 2013*, pp. 181-188.
[9] A general purpose C++ library Dlib. URL: http://dlib.net.
[10] O. S. Zaikin, S. E. Kochemazov. “An Improved SAT-Based Guess-and-Determine Attack on the Alternating Step Generator,” In *Proc. International Conference on Information Security*, Lecture Notes in Computer Science, vol. 10599, 2017, pp. 21–38.
[11] A. A. Semenov, O. S. Zaikin, D. V. Bespalov, and M. A. Posypkin, “Parallel Logical Cryptanalysis of the Generator A5/1 in BNB-Grid System,” In *Proc. Parallel computing technologies (PaCT)*, Lecture Notes in Computer Science, vol. 6873, 2011, pp 473-483.
[12] O. S. Zaikin, “Application of parallel SAT solving algorithms for cryptanalysis of the shrinking and self-shrinking keystream generators,” *International Journal of Open Information Technologies*, vol. 6 (10), 2018, pp. 29-33.
[13] I. V. Otpuschennikov, A. A. Semenov, I. A. Gribanova, O. S. Zaikin, and S. E. Kochemazov. “Encoding cryptographic functions to SAT using Transalg system,” In *Proc. ECAI’2016*, Frontiers in Artificial Intelligence and Applications, vol. 285, 2016, pp. 1594-1595.
[14] Irkutsk supercomputing center SB RAS. URL: http://hpc.icc.ru.

Using support vector machine to evaluate usefulness of conflict clauses in CDCL derivation

Oleg S. Zaikin, Stepan E. Kochemazov

***Abstract*—Many state-of-the-art algorithms for solving Boolean satisfiability problem (SAT) are based on the CDCL algorithm. CDCL generates a lot of so-called conflict clauses that correspond to traversed branches of a tree of possible solutions. To maintain high speed of CDCL-based algorithms, it is required to periodically remove some conflict clauses. Therefore, the problem of evaluating conflict clauses usefulness arises. In the present study, a heuristic for solving this problem is proposed that is based on support vector machines. On the first stage, a family of simplified versions of an original SAT instance is constructed, then they are solved via some SAT solver. On the second stage, a support vector machine is trained. During this process, a conflict clause is considered useful if it is not removed at the time of finding a solution of at least one simplified subproblem. On the third stage, an original SAT instance is solved, while the usefulness of some conflict clauses is evaluated by the trained support vector machine. Based on the proposed heuristic, a modified version of a state-of-the-art CDCL solver is implemented. According to the computational experiments, the modified version is more efficient on a few families of hard SAT instances.**

***Keywords*—SAT, CDCL, support vector machine.**

REFERENCES

[1] M. R. Garey, D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co. New York, NY, USA, 1979.

[2] J. P. Marques-Silva, K. A. Sakallah, "GRASP-a new search algorithm for satisfiability," in *Proc. IEEE International Conference on Computer-Aided Design (ICCAD)*, 1996, pp. 220–227.

[3] J. H. Liang, V. Ganesh, P. Poupart, and K. Czarnecki, "Learning rate based branching heuristic for SAT solvers," in *Proc. Theory and Applications of Satisfiability Testing (SAT)*, Lecture Notes in Computer Science, vol. 9710, 2016, pp 123-140.

[4] M. N. Katehakis, A. F. Veinott, "The Multi-Armed Bandit Problem: Decomposition and Computation," *Mathematics of Operations Research*, vol. 12 (2), 1995, pp. 262–268.

[5] C. Corinna, V. N. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20 (3), 1995, pp. 273–297.

[6] O. S. Zaikin, "A parallel SAT solving algorithm based on improved handling of conflict clauses," In *Proc. International young scientists conference in high performance computing and simulation*, Procedia computer science, vol. 12083, 2017, pp. 103-111.

[7] G. Audemard, L. Simon, "On the Glucose SAT solver," *International Journal on Artificial Intelligence Tools*, vol. 27 (1), 2018, pp. 1-25.

[8] M. Heule, W. A. Hunt Jr., and N. Wetzler, "Trimming while checking clausal proofs," In *Proc. FMCAD 2013*, pp. 181-188.

[9] A general purpose C++ library Dlib. URL: http://dlib.net.

[10] O. S. Zaikin, S. E. Kochemazov. "An Improved SAT-Based Guess-and-Determine Attack on the Alternating Step Generator," In *Proc. International Conference on Information Security*, Lecture Notes in Computer Science, vol. 10599, 2017, pp. 21–38.

[11] A. A. Semenov, O. S. Zaikin, D. V. Bespalov, and M. A. Posypkin, "Parallel Logical Cryptanalysis of the Generator A5/1 in BNB-Grid System," In *Proc. Parallel computing technologies (PaCT)*, Lecture Notes in Computer Science, vol. 6873, 2011, pp 473-483.

[12] O. S. Zaikin, "Application of parallel SAT solving algorithms for cryptanalysis of the shrinking and self-shrinking keystream generators," *International Journal of Open Information Technologies*, vol. 6 (10), 2018, pp. 29-33.

[13] I. V. Otpuschennikov, A. A. Semenov, I. A. Gribanova, O. S. Zaikin, and S. E. Kochemazov. "Encoding cryptographic functions to SAT using Transalg system," In *Proc. ECAI'2016*, Frontiers in Artificial Intelligence and Applications, vol. 285, 2016, pp. 1594-1595.

[14] Irkutsk supercomputing center SB RAS. URL: http://hpc.icc.ru.