УДК 519.7
ББК В 22.1

ПРОЦЕДУРЫ ПОСТРОЕНИЯ ДЕКОМПОЗИЦИОННЫХ МНОЖЕСТВ ДЛЯ РАСПРЕДЕЛЕННОГО РЕШЕНИЯ SAT-ЗАДАЧ В ПРОЕКТЕ ДОБРОВОЛЬНЫХ ВЫЧИСЛЕНИЙ SAT@HOME[1]

Заикин О. С.[2], Семенов А. А.[3]
(*ФГБУН Институт динамики систем и теории управления СО РАН, Иркутск*)
Посыпкин М. А.[4]
(*ФГБУН Институт проблем передачи информации РАН, Москва*)

В статье предложен новый подход к построению декомпозиционных множеств, используемых для крупноблочного распараллеливания SAT-задач и их решения в распределенных вычислительных средах. Предложенные алгоритмы используется в проекте добровольных распределенных вычислений SAT@home.

Ключевые слова: Выполнимость булевых формул, метод имитации отжига, добровольные распределенные вычисления, SAT@home.

---

[1] *Работа выполнена при финансовой поддержке РФФИ, гранты №№ 11-07-00377-а, 11-07-00428-а, 13-07-00291-а, 13-07-00768-а, и Совета по грантам Президента РФ для поддержки молодых ученых* (*стипендия СП-1855.2012.5*).

[2] *Олег Сергеевич Заикин, научный сотрудник, кандидат технических наук* (*zaikin.icc@gmail.com*).

[3] *Александр Анатольевич Семенов, заведующий лабораторией, кандидат технических наук, доцент* (*biclop.rambler@yandex.ru*).

[4] *Михаил Анатольевич Посыпкин, ведущий научный сотрудник, кандидат физико-математических наук, доцент* (*mposypkin@gmail.com*).

1. Введение

Обширный класс задач современной кибернетики можно рассматривать в контексте общей проблемы поиска решений булевых уравнений. Задачи поиска решений уравнений вида КНФ = 1 (КНФ – конъюнктивная нормальная форма) называются *SAT*-задачами [14]. Спектр применения *SAT*-подхода очень широк – на сегодня известно множество работ, в которых различные комбинаторные проблемы ставятся и решаются в форме *SAT*-задач. Сказанное касается верификации, криптографии, комбинаторики, биоинформатики и других областей. Все известные алгоритмы решения *SAT*-задач экспоненциальны в худшем случае (*SAT*-проблемы *NP*-трудны в общей постановке). Однако современные *SAT*-решатели успешно справляются с обширными классами «индустриальных» тестов, в основе которых лежат задачи из перечисленных выше областей. Повышение эффективности решения *SAT*-задач, в том числе разработка алгоритмов, работающих в параллельных и распределенных вычислительных средах, является практически важным и актуальным направлением исследований.

При разработке вычислительных алгоритмов, применяемых к решению *NP*-трудных задач, принципиальным является вопрос аргументации эффективности таких алгоритмов. Если предлагаемые алгоритмы показывают хорошие результаты на аргументированно трудных тестах, то разумно предполагать, что они будут применимы и к задачам, вычислительная трудность в которые не заложена искусственно. В настоящей работе в качестве аргументированно трудных тестов используются *SAT*-задачи, кодирующие проблемы криптоанализа ряда систем шифрования.

Далее мы представим новый подход к крупноблочному распараллеливанию *SAT*-задач, применяемый в проекте добровольных распределенных вычислений *SAT@home*. Проект *SAT@home* [2, 6, 26] создан в ИДСТУ СО РАН в сотрудничестве с ИППИ РАН и представляет собой систему добровольных распределенных вычислений на платформе *BOINC* [13], предназначенную для решения задачи о булевой выполнимости (*SAT*). Данный проект был запущен 29.09.2011г. С 21.12.2011г. по 07.05.2012г. в *SAT@home* решались задачи криптоанализа известного генератора ключевого потока A5/1.

Приведем краткий план статьи. В следующем разделе будут в общих чертах описаны техники сведения различных комбинаторных проблем к *SAT*-задачам. Третий раздел содержит описание как известных, так и новых методов крупноблочного распараллеливания *SAT*-задач, ориентированных на использование в распределенных вычислительных средах. В четвертом разделе кратко описан проект добровольных распределенных вычислений *SAT@home*. Пятый раздел содержит результаты вычислительных экспериментов.

2. Сведение комбинаторных проблем к SAT-задачам

Теоретически возможность эффективного (за полиномиальное время) преобразования проблемы распознавания произвольного языка из класса *NP* в проблему распознавания языка выполнимых КНФ есть следствие теоремы С. Кука [16]. Используя общие идеи С. Кука, можно сводить к проблеме поиска набора, выполняющего КНФ, разнообразные комбинаторные задачи. Смысл этих действий в том, что для решения *SAT*-задач существуют программные разработки, называемые *SAT*-решателями, которые весьма хорошо себя зарекомендовали за последние несколько лет [7]. Подавляющее большинство современных эффективных *SAT*-решателей построено на базе алгоритма *DPLL* [17, 25].

Известно множество различных примеров преобразования комбинаторных задач в *SAT*-задачи [27]. В ситуациях, когда исходная задача является задачей обращения дискретной функции (к данному типу относятся задачи криптоанализа), можно использовать системы автоматической трансляции процедурных описаний функций в виде программ в системы булевых уравнений и *SAT*-задачи (например, [4]). Общий принцип такого рода трансляторов состоит в следующем.

Пусть дана дискретная функция

$$f:\{0,1\}^* \to \{0,1\}^*,$$

определенная всюду на $\{0, 1\}^*$ и вычислимая за полиномиальное время программой T_f для детерминированной машины Тьюринга. Данная программа естественным образом задает счетное семейство функций

$$f_k:\{0,1\}^k \to \{0,1\}^*, k \in \mathrm{N}.$$

Каждую функцию f_k из данного семейства можно реализовать схемой $S(f_k)$ из функциональных элементов некоторого полного базиса, например, $\{\&, \neg\}$. Функция роста размера (т.е. числа функциональных элементов) получаемого семейства схем будет ограничена сверху полиномом от k. Рассматриваем схему $S(f_k)$ как направленный граф, в множестве вершин которого выделены k входных. Входным вершинам сопоставляются булевы переменные $x_1, \ldots, x_k$. Пусть g – произвольная внутренняя вершина схемы $S(f_k)$. Данной вершине соответствует функциональный элемент $E(g) \in \{\&, \neg\}$. Сопоставим вершине g новую булеву переменную $v(g)$ и формулу $F(g)$, которая имеет вид либо $v(g) \equiv u \,\&\, w$, если $E(g)$ – это «&», либо $v(g) \equiv \overline{u}$, если $E(g)$ – это «¬». Здесь u, w – булевы переменные, соответствующие входам элемента g, т.е. родителям вершины g в графе, представляющем схему $S(f_k)$. Пусть $C(g)$ – КНФ-представление формулы $F(g)$. КНФ, кодирующая схему $S(f_k)$, имеет следующий вид:

$$\underset{g \in S(f_k)}{\&} C(g).$$

Тогда КНФ, кодирующая задачу обращения функции f_k в точке $y \in Range\, f_k$, $y = (\sigma_1, \ldots, \sigma_m)$, $\sigma_j \in \{0, 1\}$, $j \in \{1, \ldots, m\}$, выглядит следующим образом:

$$C(f_k, y) = \left(\underset{g \in S(f_k)}{\&} C(g)\right) \cdot y_1^{\sigma_1} \cdot \ldots \cdot y_m^{\sigma_m},$$

где

$$z^\sigma = \begin{cases} z, \sigma = 1, \\ \overline{z}, \sigma = 0; \end{cases}$$

y_j, $j \in \{1, \ldots, m\}$ – булевы переменные, соответствующие выходам схемы $S(f_k)$. Используя идеи работы [11], несложно показать, что решив булево уравнение $C(f_k, y) = 1$, мы можем найти такое слово $x \in \{0, 1\}^k$, что $f_k(x) = y$.

Подобный подход используется в системе *Transalg* [4], которая предназначена для трансляции в булевы уравнения программ, написанных на специальном *C*-подобном языке. Все рассматриваемые в настоящей работе *SAT*-задачи, кодирующие задачи обращения дискретных функций, построены при помощи этой системы.

3. Алгоритмы крупноблочного распараллеливания SAT-задач

Как уже говорилось, *SAT*-подход можно применять к решению комбинаторных задач из весьма широкого класса. В связи с этим актуальна проблема построения для решения *SAT*-задач алгоритмов, работающих в параллельных и распределенных вычислительных средах. За последние 5 лет появился ряд *SAT*-решателей, использующих обмен накапливаемыми конфликтными ограничениями между параллельно работающими вычислительными узлами. Большинство таких решателей являются многопоточными приложениями [19, 20, 28] и обычно задействуют небольшое число ядер. Известные *MPI*-решатели применимы лишь к некоторым ограниченным классам задач [23]. В силу высокой вычислительной трудности некоторых *SAT*-задач особую актуальность приобретает проблема разработки распределенных алгоритмов, работающих в распределенных средах со слабым взаимодействием между узлами. Исследования по данной тематике стали появляться совсем недавно. Отметим статью [29], в которой описан распределенный *SAT*-решатель, работающий в *peer-to-peer*-сетях, а также работы [21, 22], содержащие опыт решения *SAT*-задач в грид-средах.

Специализированная грид-среда, в которой решалась *SAT*-задача, кодирующая криптонализ широко известного генератора поточного шифрования A5/1, была описана в [5, 30]. В данных работах использовалась техника распараллеливания *SAT*-задач, предложенная в статье [3]. Далее мы в общих чертах описываем подход работы [3] и развиваем его в направлении построения автоматических процедур поиска декомпозиционных множеств с «хорошими» свойствами.

Итак, пусть нам дана произвольная *SAT*-задача в виде уравнения $C = 1$, где C – конъюнктивная нормальная форма над множеством булевых переменных $X = \{x_1, \ldots, x_n\}$. Произвольное множество $\widetilde{X} = \{x_{i_1}, \ldots, x_{i_d}\}$, $\widetilde{X} \subseteq X$, назовем декомпозиционным множеством. Пусть $Y \in \{0, 1\}^d$ – произвольный набор значений истинности переменных из $\widetilde{X}$. Через $C|_Y$ обозначается КНФ, полученная в результате подстановки в C набора Y. Множество $\Delta(C, \widetilde{X}) = \{C|_Y\}_{Y \in \{0,1\}^d}$ называется декомпозиционным семейством для исходной *SAT*-задачи. Очевидно, что решив все *SAT*-задачи из $\Delta(C, \widetilde{X})$, мы получим решение исходной *SAT*-задачи. Обработку множества $\Delta(C, \widetilde{X})$ можно осуществлять в распределенной вычислительной среде. Однако при различных альтернативах $\Delta(C, \widetilde{X})$ мы будем получать различное время его обработки. Очевидно, что в данной ситуации необходим компромисс между числом *SAT*-задач в $\Delta(C, \widetilde{X})$ и их сложностью. Для построения таких компромиссных декомпозиционных множеств в [3] был предложен метод прогнозных функций. Значением прогнозной функции на конкретном $\widetilde{X}$ является прогноз общей трудоемкости обработки множества $\Delta(C, \widetilde{X})$. Вычисляется это значение следующим образом: случайно из $\{0, 1\}^d$ выбираются векторы $Y_1, \ldots, Y_Q$, где Q – число, такое что $Q \ll 2^d$. Строится семейство КНФ $\{C|_{Y_1}, \ldots, C|_{Y_Q}\}$, которое обрабатывается *SAT*-решателем S (на рабочей станции или вычислительном кластере). Пусть t – время обработки данной случайной выборки. Тогда $F(C, \widetilde{X}) = 2^d \cdot \dfrac{t}{Q}$ – значение прогнозной функции на $\widetilde{X}$.

Несложно понять, что всегда можно эффективно вычислить некоторое стартовое значение прогнозной функции. Действительно, это справедливо, например, если $\tilde{X} = X$. Если мы решаем *SAT*-задачу, кодирующую обращение некоторой дискретной функции, и в роли *S* используем решатель на базе алгоритма *DPLL*, то в качестве стартового декомпозиционного множества можно выбрать ядро *DPLL*-вывода [8]. Значение прогнозной функции на этом множестве также подсчитывается эффективно.

В работах [3, 5, 30] была использована весьма простая стратегия улучшения значений прогнозной функции. А именно, в качестве стартового декомпозиционного множества $\tilde{X}_1$ выбиралось ядро *DPLL*-вывода (поскольку рассматривались только задачи обращения дискретных функций). Далее делались попытки улучшить значение прогнозной функции на множествах следующего вида:

$$\tilde{X}_1 \supset \tilde{X}_2 \supset \ldots \supset \tilde{X}_r \supset \tilde{X}_* \supset \ldots \supset \tilde{X}_s ,$$

причем для всех $k \in \{1, \ldots, s - 1\}$ выполнялось $|\tilde{X}_{k+1}| = |\tilde{X}_k| - 1$. Здесь $\tilde{X}_*$ – множество с наилучшим значением прогнозной функции среди множеств $\tilde{X}_1, \ldots, \tilde{X}_s$. Оно и выбиралось на роль декомпозиционного множества для решения рассматриваемой *SAT*-задачи в распределенной вычислительной среде.

Описанная стратегия, несмотря на свою простоту, дала неплохие результаты в криптоанализе некоторых генераторов ключевого потока. Однако она оказалась бесполезной при решении задачи криптоанализа генератора A5/1. Декомпозиционное множество для *SAT*-задачи, кодирующей криптоанализ данного генератора, фактически было найдено «вручную» [30]. Это множество изображено на рис. 1 – булевы переменные, включаемые в множество, кодируют значения ячеек, выделенных серой заливкой (описание генератора A5/1 взято из статьи [15]).

Далее мы рассматриваем общую задачу поиска декомпозиционного множества с наилучшим значением прогнозной функции как задачу глобальной оптимизации на конечном множестве и предлагаем для ее решения новую стратегию.

Итак, рассматриваем *SAT*-задачу $C = 1$, где C – КНФ над множеством булевых переменных $X = \{x_1, \ldots, x_n\}$. Произвольное множество $\widetilde{X} \subset X$ будем задавать характеристическим вектором $\alpha^{\widetilde{X}} = \left(\alpha_1^{\widetilde{X}}, \ldots, \alpha_n^{\widetilde{X}}\right)$, таким что

$$\alpha_i^{\widetilde{X}} = \begin{cases} 1, \text{ если } x_i \in \widetilde{X}, \\ 0, \text{ если } x_i \notin \widetilde{X}. \end{cases}$$

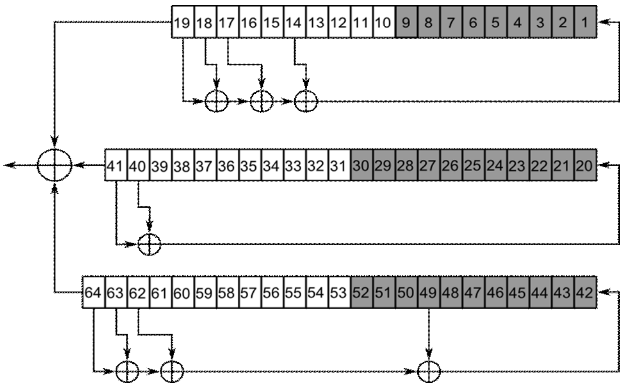

Рис. 1. Декомпозиционное множество, структура которого обусловлена особенностями алгоритма генератора A5/1 [*30*]

Значениями аргумента прогнозной функции $\Phi(\cdot)$ являются всевозможные векторы вида $\alpha^{\widetilde{X}}$. Полагаем, что $\Phi\left(\alpha^{\widetilde{X}}\right) = F\left(C, \widetilde{X}\right)$ (определение функции $F\left(\widetilde{X}, C\right)$ дано выше). Переход к векторам $\alpha^{\widetilde{X}}$ нам нужен лишь для того, чтобы корректно определять окрестности точек в пространстве поиска (очевидно, что таким пространством является множество 2^X). После этого мы можем использовать любую схему поиска глобального минимума функции $\Phi(\cdot)$ на 2^X.

Приведем здесь ряд аргументов, обосновывающих использование схемы, описание которой приведено ниже. Во-первых, отметим, что функция $\Phi(\cdot)$ не задана аналитически – ее значения есть величины, наблюдаемые во времени, фактически это реакция вычислительной среды на выбранную форму распараллеливания (вид декомпозиционного множества). В связи с этим невозможно использовать для оптимизации $\Phi(\cdot)$ «традиционные» методы, так или иначе привлекающие «аналитические» свойства рассматриваемой функции (гладкость, выпуклость, *d.c.* [10], и др.). В данной ситуации нам представляется оправданным применить для решения описанной проблемы метаэвристические алгоритмы. В частности, на данном этапе реализована вычислительная схема имитации отжига [24]. Вторая особенность рассматриваемой задачи состоит в том, что далеко не все значения $\Phi(\cdot)$ необходимо досчитывать. Действительно, если Φ' – некоторое рекордное значение данной функции на данный момент времени, и для точки $\alpha^{\widetilde{X}}$ можно сделать вывод, что $\Phi\left(\alpha^{\widetilde{X}}\right) > \Phi'$, то очевидно, что дальнейшее вычисление значения $\Phi\left(\alpha^{\widetilde{X}}\right)$ можно прервать и перейти к следующей точке либо принять точку $\alpha^{\widetilde{X}}$ за центр новой окрестности с некоторой вероятностью.

В соответствии со схемой имитации отжига [24], минимизация функции $\Phi(\cdot)$ рассматривается как итеративный процесс переходов между точками пространства поиска:

$$\alpha^0 \to \alpha^1 \to \ldots \to \alpha^i \to \ldots \to \alpha^* .$$

Переход от α^i к α^{i+1} осуществляется в два этапа. Сначала к α^i применяется вероятностное преобразование, результатом которого является точка $\widetilde{\alpha}^i$ из некоторой окрестности α^i. Точка $\widetilde{\alpha}^i$ становится точкой α^{i+1} с вероятностью, обозначаемой $\Pr\left\{\widetilde{\alpha}^i \to \alpha^{i+1} \mid \alpha^i\right\}$. Данная вероятность задается следующим образом:

$$\Pr\left\{\widetilde{\alpha}^i \to \alpha^{i+1} \mid \alpha^i\right\} = \begin{cases} 1, \text{ если } \Phi\left(\widetilde{\alpha}^i\right) < \Phi\left(\alpha^i\right), \\ \exp\left(-\dfrac{\Phi\left(\widetilde{\alpha}^i\right) - \Phi\left(\alpha^i\right)}{T_i}\right), \text{ если } \Phi\left(\widetilde{\alpha}^i\right) \geq \Phi\left(\alpha^i\right). \end{cases}$$

Изменение параметра T_i соответствует уменьшению «температуры кристаллизирующейся среды». Обычно полагают, что $T_i = Q \cdot T_{i-1}$, $i \geq 1$, где $Q \in (0, 1)$. Процесс стартует при некотором начальном значении T_0 и продолжается до тех пор, пока «температура» не станет меньше заданного порогового значения T_{inf}.

Описанная схема минимизации прогнозных функций была реализована в виде параллельного *MPI*-приложения, работающего на вычислительном кластере. Во время вычислений приложение отслеживает ситуации превышения рекордных значений прогнозной функции с последующим прерыванием соответствующего вычисления либо переходом к новой точке. В целом данный процесс работает в соответствии со схемой, описанной в [1] (используются неблокирующие обмены в *MPI*-среде). Результаты вычислительных экспериментов приведены в разделе 5.

4. Проект добровольных распределенных вычислений SAT@home

4.1. ОБЩИЕ СВЕДЕНИЯ О ДОБРОВОЛЬНЫХ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛЕНИЯХ

Здесь мы кратко коснемся общей идеологии добровольных вычислений. Первыми добровольными проектами были *GIMPS* и *distributed.net*, запущенные соответственно в 1996 и 1997 годах. Проект *GIMPS* был направлен на поиск чисел Мерсенна, а *distributed.net* - на решение различных вариантов задачи криптоанализа шифра *RC*5. В 1996 году на одной из конференций по радиоастрономии была представлена концепция проекта *SETI@home*, предназначенного для обработки интенсивных потоков данных, поступающих от мощных радиотелескопов. Данный проект был запущен в 1999 году, а в 2002 году на его основе была разработана открытая платформа *BOINC* [13]. И если изначально для создания добровольных проектов требовались ресурсы больших научных коллективов, то с использованием *BOINC* построение каждого нового проекта стало вполне по силам небольшим лабораториями и даже отдельным энтузиастам. Из 70 активных на данный момент проектов добровольных вычислений 65 построены на платформе *BOINC*.

Проект на платформе *BOINC* состоит из следующих основных частей: серверного ПО, веб-сайта и прикладного ПО. В состав серверного ПО входят следующие службы:

– *work_generator* – создает задания для обработки;

– *validator* – проверяет корректность присланных с ПК пользователей результатов, а также начисляет кредиты за корректные результаты;

– *assimilator* – обрабатывает корректные результаты.

Сайт проекта содержит следующую информацию: цели проводимых исследований, полученные результаты, список публикаций авторов проекта, производительность проекта, рейтинг лучших (в смысле количества набранных в проекте кредитов) участников и т.п. Также участнику доступен форум, на котором идет общение с разработчиками проекта.

Прикладное ПО представлено набором исполняемых файлов для различных типов операционных систем (ОС) и процессоров. Для получения значительных вычислительных ресурсов в проекте следует поддерживать версии для основных семейств ОС (*Windows*, *Linux*, *Mac*) и процессоров (x86, x64).

4.2. ПРОЕКТ ДОБРОВОЛЬНЫХ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛЕНИЙ SAT@HOME

Данный проект был запущен 29 сентября 2011 года. *SAT@home* [6] – совместный проект Института динамики систем и теории управления СО РАН и Института проблем передачи информации РАН. При создании проекта была использована открытая платформа *BOINC*.

Перед использованием проекта SAT@home выполняется подготовительный этап. На этом этапе решатель *PD-SAT* [1], запущенный на вычислительном кластере, находит для исходной *SAT*-задачи некоторое декомпозиционное множество в соответствии с описанными выше алгоритмами минимизации прогнозных функций. Найденное декомпозиционное множество передается серверному ПО проекта вместе с исходной КНФ в качестве входных данных.

Схема работы проекта представлена на рис. 2. Серверное ПО отвечает за создание заданий в базе данных проекта, а также за обработку результатов выполнения заданий, присылаемых с ПК пользователей. Отправкой заданий на ПК пользователей и получением результатов занимаются стандартные службы *BOINC*.

Клиентское приложение в виде исполняемых файлов для конкретной операционной системы запускает на ПК пользователей стандартный *BOINC*-клиент. Основу приложения, запускаемого на ПК пользователей, составляют *SAT*-решатели *minisat*-1.14.1 и *minisat*-2.0 [18], модифицированные с учетом особенностей решаемых в проекте *SAT*-задач. В результате решения в общем случае всех подзадач находится решение исходной *SAT*-задачи.

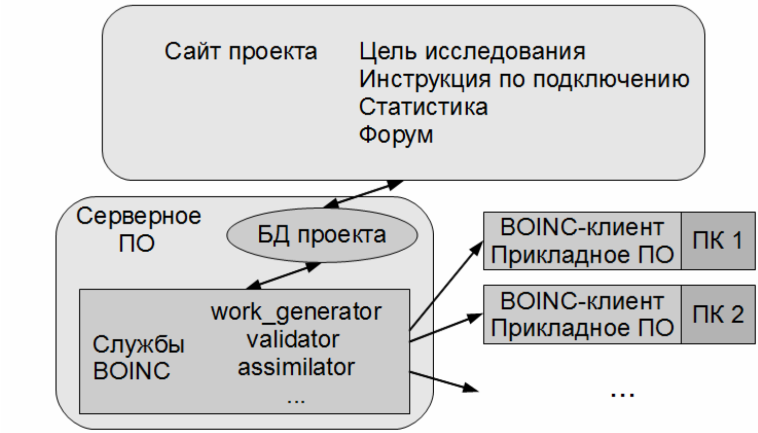

Рис. 2. Схема проекта SAT@home

По состоянию на 19 апреля 2013 г. *SAT@home* имеет следующие характеристики:

– 2936 активных ПК, около 80 % под управлением ОС семейства *Windows*;

– версии клиентского приложения: *windows* x86, *linux* x86, linux x64;

– средняя реальная производительность 2,2 терафлопс, максимальная 6,3 терафлопс.

5. Вычислительные эксперименты

В данном разделе мы приводим результаты тестирования описанных выше процедур минимизации прогнозных функций, а также результаты решения реальных задач в проекте *SAT@home*. На рис. 1 приведено декомпозиционное множество, использованное для решения задачи криптоанализа известного генератора поточного шифрования A5/1. Наиболее успешным методом криптоанализа данного генератора является так называемый «*rainbow*»-метод. Однако известные *rainbow*-таблицы [12] покрывают ключевое пространство A5/1 примерно на 88% и не дают результатов (при реалистичных предположениях на условия криптоанализа) для тестов, в которых используются оставшиеся 12% ключей. Для решения в рамках проекта *SAT@home* были построены 10 таких тестов [6] (раздел «найденные решения»). Все они были решены за полгода работы проекта (с 21.12.2011г. по 07.05.2012 г.).

Как уже отмечалось выше, декомпозиционное множество для *SAT*-задачи, кодирующей криптоанализ A5/1, фактически вычислялось «вручную». Использованные при этом соображения основаны на известных особенностях алгоритма A5/1. В частности, относительно некоторых переменных в кодирующей КНФ можно сделать вывод об их более сильном приоритете перед остальными при включении в декомпозиционное множество. Это обусловлено так называемой «функциональной семантикой» проблемы – в задачах обращения функций стартовым декомпозиционным множеством может быть множество всех переменных, кодирующих входные значения схемы, реализующей рассматриваемую функцию. Однако далеко не для всех комбинаторных задач можно делать подобные выводы. В этих ситуациях, конечно же, при построении декомпозиционных множеств необходимо использовать автоматические процедуры.

Описанный выше метод минимизации прогнозных функций является такой процедурой.

При тестировании метода на адекватность необходимо сравнение выдаваемых им результатов с некоторыми эталонами. В качестве такого эталона мы рассмотрели декомпозиционное множество для *SAT*-задачи, кодирующей криптоанализ A5/1 (рис. 1). На рис. 4 приведена структура декомпозиционного множества, найденного при помощи представленной выше процедуры минимизации прогнозных функций. При этом были использованы следующие параметры схемы имитации отжига: величина начальной «температуры» T_0 выбиралась равной 5% от значения прогнозной функции в начальной точке (выбранное случайным образом множество мощности 40 среди 64 булевых переменных, кодирующих вход генератора A5/1); конечная «температура» $T_{\inf} = 30000$.

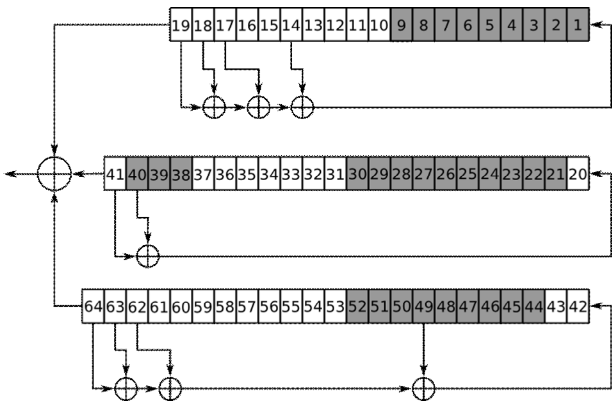

Рис. 3. Декомпозиционное множество, построенное автоматически (без привязки к особенностям исходной задачи) при помощи метода, описанного в настоящей работе

Из рис. 1 и 3 можно сделать вывод, что предложенная схема строит декомпозиционное множество, структура которого очень близка к структуре эталонного множества – метод находит множество той же мощности, что и эталонное, в котором лишь три переменные находятся «не на своем месте». Данный эффект наблюдался и в применении к другим криптографическим задачам (криптоанализ суммирующего и порогового генераторов). Это позволяет надеяться, что описанный подход позволит находить адекватные декомпозиционные множества в *SAT*-задачах, кодирующих не только задачи обращения функций, но и различные комбинаторные проблемы с невыраженной функциональной семантикой.

6. Заключение

В работе предложен новый подход к построению декомпозиционных множеств для крупноблочного распараллеливания *SAT*-задач. Показано, что декомпозиционное множество, построенное предложенным алгоритмом для *SAT*-задачи, кодирующей криптоанализ генератора A5/1, совсем незначительно отличается от известного эталонного множества. Разработанные алгоритмы предполагается использовать для решения различных комбинаторных задач в проекте распределенных вычислений *SAT@home*. В частности, предполагается использовать данный проект для поиска новых ортогональных систем латинских квадратов порядков 9 и 10.

Литература

1. ЗАИКИН О.С. *Реализация процедур прогнозирования трудоемкости параллельного решения SAT-задач* // Вестник УГАТУ. – 2010. – Т.14, №4. – С. 210–220.
2. ЗАИКИН О.С., ПОСЫПКИН М.А., СЕМЕНОВ А.А., ХРАПОВ Н.П. *Организация добровольных вычислений на платформе BOINC на примере проектов OPTIMA@home и SAT@home* // CAD/CAM/CAE Observer. – 2012. – №3(71). – С. 87–92.
3. ЗАИКИН О.С., СЕМЕНОВ А.А. *Технология крупноблочного параллелизма в SAT-задачах* // Проблемы управления. – 2008. – №1. – С. 43–50.

4. ОТПУЩЕННИКОВ И.В., СЕМЕНОВ А.А. *Технология трансляции комбинаторных проблем в булевы уравнения* // Прикладная дискретная математика. – 2011. – № 1. – С. 96–115.
5. ПОСЫПКИН М.А., ЗАИКИН О.С., БЕСПАЛОВ Д.В., СЕМЕНОВ А.А. *Решение задач криптоанализа поточных шифров в распределенных вычислительных средах* // Труды ИСА РАН. – 2009. – №46. – С. 119–137.
6. *Проект добровольных распределенных вычислений SAT@home*. - URL: http://sat.isa.ru/pdsat/ (дата обращения 19.04.2013).
7. *Сайт, посвященный задаче о булевой выполнимости*. – URL: http://www.satlive.org (дата обращения 19.04.2013).
8. СЕМЕНОВ А.А. *Декомпозиционные представления логических уравнений в задачах обращения дискретных функций* // Известия РАН. Теория и системы управления. – 2009. – №5. – С. 47–61.
9. *Список активных проектов на платформе BOINC*. - URL: http://boinc.berkeley.edu/projects.php (дата обращения 19.04.2013).
10. СТРЕКАЛОВСКИЙ А.С. *Элементы невыпуклой оптимизации*. – Новосибирск, «Наука», 2003. – 355 с.
11. ЦЕЙТИН Г.С. *О сложности вывода в исчислении высказываний* // Записки научных семинаров ЛОМИ АН СССР. – 1968. – Т. 8. – С. 234–259.
12. *A5/1 Cracking project*. - URL: http://reflextor.com/trac/a51/ (дата обращения 19.04.2013).
13. ANDERSON D.P. *BOINC: A System for Public-Resource Computing and Storage* // Proc. Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04). – P. 4–10.
14. BIERE A., HEULE V., VAN MAAREN H., WALSH T. *Handbook of Satisfiability*. - IOS Press, 2009. – 980 p.
15. BIRYUKOV A., SHAMIR A., WAGNER D. *Real time cryptanalysis of A5/1 on a PC* // Lecture Notes in Computer Science. – 2001. – Vol. 1978. – P. 1–18.

16. COOK S.A. *The complexity of theorem-proving procedures* // Third Annual ACM Symposium on Theory of Computing , Ohio, USA. – 1971. – P. 151–159.
17. DAVIS M., LOGEMANN G., LOVELAND D. *A machine program for theorem proving* // Communication of the ACM. – 1962. – Vol. 5, Issue 7. – P. 394–397.
18. EEN N., SORENSSON N. *An Extensible SAT-solver*. Lecture Notes in Computer Science. – 2003. – Vol. 2919. – P. 502–518.
19. GIL L., FLORES P., SILVEIRA L.M. *PMSat: a parallel version of MiniSAT* // Journal on Satisfiability, Boolean Modeling and Computation. – 2009. – Vol. 6. – P. 71–98.
20. HAMADI Y., JABBOUR S., SAIS L. *ManySAT: a Parallel SAT Solver* // Journal on Satisfiability, Boolean Modeling and Computation. – 2009. – Vol. 6. – P. 245–262.
21. HEULE M., KULLMANN O., WIERINGA S., BIERE A. *Cube and Conquer: Guiding CDCL SAT Solvers by Lookaheads* // Proc. Seventh International Haifa Verification Conference (HVC-11). – 2011. – P. 50–65.
22. HYVARINEN A., NIEMELA I., JUNTTILA T. *Grid-Based SAT Solving with Iterative Partitioning and Clause Learning* // Lecture Notes in Computer Science. – 2011. –Vol. 6876. – P. 385–399.
23. IGNATIEV A., SEMENOV A. *DPLL+ROBDD Derivation Applied to Inversion of Some Cryptographic Functions* // Lecture Notes in Computer Science. – 2011. – Vol. 6695. – P. 76–89.
24. KIRKPATRICK S., GELATT C.D., VECCHI M.P. *Optimization by simulated annealing* // Science. – 1983. – Vol. 220. – P. 671–680.
25. MARQUES-SILVA J.P., SAKALLAH K.A. *GRASP: A search algorithm for propositional satisfiability* // IEEE Transactions on Computers. – 1999. – Vol. 48, №5. – P. 506–521.
26. POSYPKIN M., SEMENOV A., ZAIKIN O. *Using BOINC desktop grid to solve large scale SAT problems* // Computer Science Journal. – 2012. – Vol. 13, №1. – P. 25–34.

27. PRESTWICH S. *CNF encodings* // In Handbook of Satisfiability (Editors: A. Biere, M.Heule, H. van Maaren, T. Walsh). – IOS Press, 2009. – P. 75–97.
28. SCHUBERT T., LEWIS M., BECKER B. *PaMiraXT: Parallel SAT Solving with Threads and Message Passing* // Journal on Satisfiability, Boolean Modeling and Computation. – 2009. – Vol. 6. – P. 203–222.
29. SCHULZ S., BLOCHINGER W. *Parallel SAT Solving on Peer-to-Peer Desktop Grids* // Journal Of Grid Computing. – 2010. – Vol. 8, № 3. – P. 443–471.
30. SEMENOV A., ZAIKIN O., BESPALOV D., POSYPKIN M. *Parallel logical cryptanalysis of the generator A5/1 in BNB-Grid system* // Lecture Notes in Computer Science. – 2011. – Vol. 6873. – P. 473–483.

CONSTRUCTING DECOMPOSITION SETS FOR DISTRIBUTED SOLUTION OF SAT PROBLEMS IN VOLUNTEER COMPUTING PROJECT SAT@HOME

Oleg Zaikin, Institute for System Dynamics and Control Theory of SB RAS, Irkutsk, researcher, Cand.Sc. (zaikin.icc@gmail.com).
Alexander Semenov, Institute for System Dynamics and Control Theory of SB RAS, Irkutsk, head of laboratory, Cand.Sc, assistant professor (biclop.rambler@yandex.ru).
Mikhail Posypkin, Institute for Information Transmission Problems of RAS, Moscow, leading researcher, Cand.Sc., assistant professor (*mposypkin@gmail.com*).

Abstract: *We suggest an approach to construct decomposition sets for coarse-grained parallelization of SAT problems. Decomposition sets are used for distributed solving of hard SAT problems. The proposed algorithms are used in the volunteer computing project SAT@home.*

Keywords: Boolean satisfiability problem, simulated annealing, volunteer computing, SAT@home.

Статья рекомендована к публикации программным комитетом международной конференции
«*Параллельные вычисления и задачи управления*» (*PACO*),
Россия, Москва, 24-26 октября 2012 г.
Поступила в редакцию 22.04.2013.
Опубликована 31.05.2013.