Серия «Математика»
2012. Т. 5, № 4. С. 79—94

Онлайн-доступ к журналу:
http://isu.ru/izvestia

ИЗВЕСТИЯ
Иркутского
государственного
университета

УДК 519.6

Алгоритмы построения декомпозиционных множеств для крупноблочного распараллеливания SAT-задач *

А. А. Семенов
ИДСТУ СО РАН

О. С. Заикин
ИДСТУ СО РАН

Аннотация. В работе приводятся алгоритмы построения декомпозиционных множеств, используемых для крупноблочного распараллеливания SAT-задач и их последующего решения в распределенных вычислительных средах. В основе предлагаемых алгоритмов лежит вычислительная схема метода Монте-Карло.

Ключевые слова: метод Монте-Карло; дискретные функции; SAT-задачи; крупноблочный параллелизм.

1. Введение

Напомним, что SAT-задачами [1] называются задачи поиска решений булевых уравнений вида КНФ=1, где КНФ — конъюнктивная нормальная форма. В своей общей постановке SAT-задачи являются NP-трудными. Однако к ним эффективно сводится обширный класс комбинаторных проблем, имеющих важные практические приложения. Поэтому разработка вычислительных алгоритмов решения SAT-задач является актуальным и интенсивно развивающимся направлением исследований. В последние несколько лет резко вырос интерес к разработке алгоритмов решения SAT-задач в параллельных и распределенных вычислительных средах. Свидетельством этого являются регулярно проводимые с 2008 года конкурсы параллельных SAT-решателей [2]. Существующие на сегодня параллельные алгоритмы решения SAT-задач

* Работа выполнена при финансовой поддержке РФФИ (грант 11-07-00377-а) и Совета по грантам Президента РФ для поддержки молодых российских ученых (проект СП-1855.2012.5).

можно условно разделить на два семейства: алгоритмы, использующие мелкозернистую (fine-grained) концепцию параллелизма, и алгоритмы, использующие крупноблочный (coarse-grained) подход. В мелкозернистых решателях происходит интенсивный обмен накапливаемыми конфликтными ограничениями между вычислительными узлами, поэтому данный подход допускает эффективную реализацию лишь в средах с быстрым межпроцессорным взаимодействием (многоядерные процессоры и кластеры) [3, 5]. При крупноблочном распараллеливании можно использовать грид-системы, скорость обмена информацией между вычислительными узлами в которых может быть крайне малой. Построение SAT-решателей, работающих в грид-системах, – сравнительно новое направление, активно развивающееся последние два года [6, 8]. Настоящая статья содержит подход к крупноблочному распараллеливанию SAT-задач, ориентированный на решение данных задач именно в грид-средах. Описанные в статье алгоритмы использовались для организации вычислений в добровольном распределенном проекте SAT@home [8].

Приведем краткий план статьи. Во втором разделе приводятся некоторые теоретические результаты, позволяющие сформировать общий взгляд на проблему крупноблочного распараллеливания SAT-задач. А именно, здесь описывается подход к оцениванию качества декомпозиционных множеств, опирающийся на идеологию метода Монте-Карло. В третьем разделе описаны алгоритмы оптимизации прогнозных функций, допускающие эффективную реализацию в многопроцессорных вычислительных средах. В последнем разделе приведены результаты численного тестирования описанных алгоритмов.

2. Методы крупноблочного распараллеливания SAT-задач

Рассматриваем уравнение вида

$$C\left(x_1,\ldots,x_n\right)=1 \tag{2.1}$$

где $C\left(x_1,\ldots,x_n\right)$ — конъюнктивная нормальная форма (КНФ) над множеством булевых переменных $X=\{x_1,\ldots,x_n\}$. Решением уравнения 2.1 называется произвольный набор $\alpha_1,\ldots,\alpha_n$, $\alpha_i\in\{0,1\}$, $i=1,\ldots,n$, такой, что подстановка $x_1=\alpha_1,\ldots,x_n=\alpha_n$ в формулу $C\left(x_1,\ldots,x_n\right)$ и применение элементарных булевых операций дает константу 1. Решить SAT-задачу, определяемую уравнением 2.1, означает найти произвольное его решение или доказать отсутствие решений.

Решать уравнение 2.1 можно при помощи различных подходов. На сегодняшний день наиболее эффективны так называемые «CDCL-решатели» (от Conflict Driven Clause Learning), базирующиеся на нехро-

нологических версиях алгоритма DPLL [9, 10]. Именно этот класс алгоритмов будет рассматриваться далее.

Поскольку SAT-задачи в общей постановке NP-трудны, то при разработке вычислительных алгоритмов для их решения востребованы различные технологии распараллеливания. Это может быть распараллеливание «на уровне алгоритма» либо распараллеливание «по данным». К первому типу можно отнести мелкозернистые стратегии, используемые в решателях [3, 5]. Применяемый в них параллелизм подразумевает обмен конфликтными дизъюнктами, параллельно накапливаемыми на различных узлах вычислительной среды, и, как следствие, требует высокой скорости межпроцессорных взаимодействий. Распараллеливание по данным обычно предполагает разбиение пространства поиска на непересекающиеся области и последующую независимую обработку этих областей в распределенной вычислительной среде. Этот подход является по своей сути крупноблочным и не требует интенсивного межпроцессорного взаимодействия, а следовательно, хорошо подходит для реализации в грид-системах. Весь дальнейший материал статьи посвящен именно крупноблочному подходу.

Один из простейших методов крупноблочного распараллеливания SAT-задач использует естественную стратегию расщепления булевых формул и состоит в следующем. Выберем некоторую переменную $x_{i_1} \in X$ и расщепим исходную КНФ $C = C(x_1, \ldots, x_n)$ на две — $C|_{x_{i_1}=0}$ и $C|_{x_{i_1}=1}$. Здесь через $C|_{x=\alpha}$ обозначена КНФ, полученная в результате подстановки в C значения $\alpha \in \{0, 1\}$ переменной x (с последующим выполнением возможных элементарных булевых операций). КНФ $C|_{x_{i_1}=0}$ и $C|_{x_{i_1}=1}$ могут быть точно так же расщеплены по некоторым переменным из $X \setminus \{x_{i_1}\}$. В результате некоторого числа шагов данного процесса будет построено двоичное дерево, которое далее будем обозначать через $T_d(C)$, полагая при этом, что каждый путь в данном дереве имеет длину d, $d \leq n$. Корню $T_d(C)$ приписана КНФ C, а всем остальным узлам приписаны КНФ вида $C|_{x_{i_1}=\alpha_{i_1},\ldots,x_{i_k}=\alpha_{i_k}}$, $k \leq d$. Несложно понять, что решив все SAT-задачи, соответствующие листьям данного дерева (часть из них могут оказаться тривиальными), мы решим и исходную SAT-задачу. Действительно, по выполняющему набору КНФ, соответствующей некоторому листу $T_d(C)$, эффективно строится набор, выполняющий исходную КНФ C, а если все КНФ, соответствующие листьям $T_d(C)$, невыполнимы, то невыполнима и C. Очевидно, что SAT-задачи, соответствующие листьям описанного дерева, можно решать на независимых узлах некоторой параллельной вычислительной среды.

Определение 1. *Описанное выше дерево $T_d(C)$ называем деревом декомпозиции КНФ C. Множество КНФ, соответствующих листьям $T_d(C)$, будем называть декомпозиционным семейством для исходной*

КНФ C и обозначать через $\Delta(C)$. Для каждой КНФ из $\Delta(C)$ вида $C|_{x_{i_1}=\alpha_{i_1},\ldots,x_{i_d}=\alpha_{i_d}}$ назовем множество $\{x_{i_1},\ldots,x_{i_d}\}$ множеством переменных декомпозиции. Если все КНФ из $\Delta(C)$ имеют одно и то же множество переменных декомпозиции $\tilde{X}$, $\tilde{X}\subseteq X$, то назовем $\Delta(C)$ регулярным, а множество $\tilde{X}$ назовем декомпозиционным множеством. В этом случае будем говорить, что декомпозиционное множество $\tilde{X}$ порождает рассматриваемое семейство, для обозначения которого используем запись $\Delta\left(C,\tilde{X}\right)$.

Утверждение 1. *Пусть C — произвольная КНФ, $T_d(C)$ – дерево ее декомпозиции, определяющее регулярное семейство $\Delta\left(C,\tilde{X}\right)$, порожденное декомпозиционным множеством $\tilde{X}$. Тогда двоичные наборы вида $(\alpha_{i_1},\ldots,\alpha_{i_d})$ по всем листьям дерева $T_d(C)$ образуют множество $\{0,1\}^d$.*

Доказательство. В силу регулярности $\Delta\left(C,\tilde{X}\right)$ все КНФ, приписанные листьям $T_d(C)$, имеют одно и то же множество переменных декомпозиции – множество $\tilde{X}=\{x_{i_1},\ldots,x_{i_d}\}$. Поскольку $T_d(C)$ – бинарное дерево, оно имеет 2^d листьев. В каждом листе $T_d(C)$ содержится информация о значениях переменных из множества $\tilde{X}$ и разным листьям соответствуют различные двоичные наборы вида $(\alpha_{i_1},\ldots,\alpha_{i_d})$. Следовательно, множество всех таких наборов – это множество $\{0,1\}^d$. □

Далее изучаются следующие вопросы. Насколько некоторое декомпозиционное семейство $\Delta(C)$ «эффективно» в плане суммарного времени решения SAT-задач для всех входящих в него КНФ? И какие семейства в этом смысле эффективнее других? На первый взгляд может показаться, что ответить на эти вопросы, не решив всех SAT-задач из $\Delta(C)$, невозможно. Тем не менее, далее мы предложим схему построения оценок такого рода. Предлагаемые алгоритмы оценивания качества декомпозиции представляют собой варианты метода Монте-Карло [11, 12] и используют сопутствующую этому методу идейную основу.

Пусть $\tilde{X}=\{x_{i_1},\ldots,x_{i_d}\}$ — произвольное подмножество X и $\Delta\left(C,\tilde{X}\right)$ — порожденное $\tilde{X}$ регулярное декомпозиционное семейство. Зафиксируем некоторый алгоритм A решения SAT-задач. Везде далее считаем, что данный алгоритм является полным, то есть заканчивает работу на произвольном входе за конечное время. Обозначим через $t_A\left(C,\tilde{X}\right)$ время, которое потребуется алгоритму A для обработки всего семейства $\Delta\left(C,\tilde{X}\right)$. Основной исследуемой далее проблемой является проблема получения численных оценок величины $t_A\left(C,\tilde{X}\right)$.

Зададим на множестве $\{0,1\}^d$ равномерное распределение. Каждому набору $(\alpha_{i_1},\ldots,\alpha_{i_d})$, выбираемому случайно из $\{0,1\}^d$, сопоставим число

$$\xi_A(\alpha_{i_1},\ldots,\alpha_{i_d}),$$

равное времени работы алгоритма A на входе $C|_{x_{i_1}=\alpha_{i_1},\ldots,x_{i_d}=\alpha_{i_d}}$. Тем самым задана случайная величина

$$\xi_A\left(C,\tilde{X}\right)=\{\,\xi_A(\alpha_{i_1},\ldots,\alpha_{i_d})\,\}_{(\alpha_{i_1},\ldots,\alpha_{i_d})\in\{0,1\}^d},$$

имеющая некоторое вероятностное распределение на $\{0,1\}^d$. Поскольку время работы A на произвольном входе конечно по предположению, то величина $\xi_A\left(C,\tilde{X}\right)$ имеет конечное математическое ожидание и конечную дисперсию. Обозначим через $\mathrm{M}\left[\xi_A\left(C,\tilde{X}\right)\right]$ математическое ожидание данной случайной величины. Несложно показать, что имеет место следующий факт.

Утверждение 2. *Справедливо соотношение*

$$t_A\left(C,\tilde{X}\right)=2^d\cdot\mathrm{M}\left[\xi_A\left(C,\tilde{X}\right)\right]. \tag{2.2}$$

Доказательство. Рассмотрим произвольное значение $\xi_A(\alpha_{i_1},\ldots,\alpha_{i_d})$ из спектра случайной величины $\xi_A\left(C,\tilde{X}\right)$. Если для любого $\left(\alpha'_{i_1},\ldots,\alpha'_{i_d}\right)$, такого, что $\left(\alpha'_{i_1},\ldots,\alpha'_{i_d}\right)\neq(\alpha_{i_1},\ldots,\alpha_{i_d})$ имеет место $\xi_A(\alpha_{i_1},\ldots,\alpha_{i_d})\neq\xi_A\left(\alpha'_{i_1},\ldots,\alpha'_{i_d}\right)$, то значению $\xi_A(\alpha_{i_1},\ldots,\alpha_{i_d})$ приписана вероятность $\frac{1}{2^d}$ (распределение на $\{0,1\}^d$ равномерное). Если же для k различных наборов $\left(\alpha^1_{i_1},\ldots,\alpha^1_{i_d}\right),\ldots,\left(\alpha^k_{i_1},\ldots,\alpha^k_{i_d}\right)$ имеет место

$$\xi_A\left(\alpha^1_{i_1},\ldots,\alpha^1_{i_d}\right)=\cdots=\xi_A\left(\alpha^k_{i_1},\ldots,\alpha^k_{i_d}\right),$$

то значению $\xi_A\left(\alpha^1_{i_1},\ldots,\alpha^1_{i_d}\right)$ приписана вероятность $\frac{k}{2^d}$. Таким образом, величина $2^d\cdot\mathrm{M}\left[\xi_A\left(C,\tilde{X}\right)\right]$ равна суммарному времени работы алгоритма A по всем КНФ вида $C|_{x_{i_1}=\alpha_{i_1},\ldots,x_{i_d}=\alpha_{i_d}}$, где набор $(\alpha_{i_1},\ldots,\alpha_{i_d})$ пробегает все множество $\{0,1\}^d$. □

Определение 2. *Декомпозиционное множество $\tilde{X}$, для которого величина 2.2 имеет наименьшее значение по всем подмножествам множества X, назовем оптимальным.*

Для вычисления величины $\mathrm{M}\left[\xi_A\left(C,\tilde{X}\right)\right]$ будем применять метод Монте-Карло [11, 12]. Напомним, что в соответствии с данным методом для приближенного вычисления математического ожидания $\mathrm{M}[\xi]$

произвольной случайной величины ξ используется вероятностный эксперимент, представляющий собой серию из N независимых наблюдений величины ξ. Если $\xi^1, \ldots, \xi^N$ – результаты соответствующих наблюдений, то при конечности математического ожидания $\mathrm{M}[\xi]$ и дисперсии $\mathrm{D}[\xi]$ справедливо следующее соотношение:

$$\Pr\left\{\left|\frac{1}{N}\cdot\sum_{j=1}^{N}\xi^j - \mathrm{M}[\xi]\right| < \frac{3\cdot\mathrm{D}[\xi]}{\sqrt{N}}\right\} \approx 0,997.$$

Данное соотношение говорит о том, что при всех сделанных предположениях величина $\frac{1}{N}\cdot\sum_{j=1}^{N}\xi^j$ хорошо приближает $\mathrm{M}[\xi]$ при достаточно большом числе наблюдений N. Заметим, что при этом N может быть существенно меньше, чем 2^d, и тем самым применение описанного подхода позволит нам использовать этап препроцессинга для оценки суммарной трудоемкости обработки декомпозиционного семейства $\Delta\left(C, \tilde{X}\right)$.

3. Алгоритмы поиска оптимальных декомпозиционных множеств

Итак, в соответствии с описанной в предыдущем пункте схемой, процесс приближенного вычисления величины 2.2 для конкретного $\tilde{X}$ выглядит следующим образом. Делается случайная выборка из $\{0,1\}^d$ наборов значений переменных, входящих в $\tilde{X}$:

$$Q\left(C, \tilde{X}\right) = \left\{\ \left(\alpha^1_{i_1}, \ldots, \alpha^1_{i_d}\right)\ , \ldots,\ \left(\alpha^N_{i_1}, \ldots, \alpha^N_{i_d}\right)\ \right\};$$

рассматриваются случайные величины

$$\xi^j = \xi_A\left(\alpha^j_{i_1}, \ldots, \alpha^j_{i_d}\right)\ ,\ j = 1, \ldots, N;$$

вычисляется величина

$$F_A\left(C, \tilde{X}\right) = 2^d \cdot \left(\frac{1}{N}\cdot\sum_{j=1}^{N}\xi^j\right). \tag{3.1}$$

При справедливости всех сделанных выше предположений значение $F_A\left(C, \tilde{X}\right)$ является хорошим приближением величины 2.2.

Функцию $F_A\left(C, \tilde{X}\right)$ далее называем прогнозной функцией. Отметим, что значения прогнозной функции можно вычислять на обычном компьютере или кластере, дополнительно необходим лишь качественный генератор случайных чисел. Тем самым мы можем от задачи

поиска оптимального декомпозиционного множества перейти к задаче поиска декомпозиционного множества $\tilde{X}_*$, такого, что для любого $\tilde{X}, \tilde{X} \subseteq X$, $F_A\left(C, \tilde{X}\right) \geq F_A\left(C, \tilde{X}_*\right)$.

Несмотря на такую, казалось бы, естественную постановку, описанная проблема имеет ряд специфических особенностей.

1) Далеко не для каждого $\tilde{X}$ значение $F_A\left(C, \tilde{X}\right)$ может быть вычислено эффективно, поскольку при малых d вычисление $F_A\left(C, \tilde{X}\right)$ сопоставимо по трудоемкости с решением исходной SAT-задачи.

2) Функция $F_A\left(C, \tilde{X}\right)$ не задана аналитически – каждое ее значение отражает реакцию вычислительной среды на соответствующий вид декомпозиции. Таким образом, для минимизации $F_A\left(C, \tilde{X}\right)$ неприменимы методы, использующие такие свойства функций как гладкость, выпуклость, d.c. [13], и вообще любые методы, использующие задание функции в виде формулы.

Итак, первая задача, которую необходимо решить, — выбор стартового декомпозиционного множества $\tilde{X}_0$. Несложно видеть, что тривиальный пример $\tilde{X}_0$ — это множество X. Однако в некоторых ситуациях $\tilde{X}_0$ может быть существенно меньше. Далее мы приводим один частный результат о структуре $\tilde{X}_0$, который используется при решении задач обращения дискретных функций как SAT-задач.

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. Программа T_f естественным образом задает счетное семейство функций

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

Задача обращения произвольной функции f_k из данного семейства в точке $y \in range\ f_k$ состоит в нахождении такого $x \in \{0,1\}^k$, что $f_k(x) = y$. Данная задача может быть эффективно (за полиномиальное от k время) преобразована в SAT-задачу за счет пропозиционального кодирования схемы $Circ\ (f_k)$ из функциональных элементов какого-либо полного базиса (например, $\{\&, \neg\}$), которая представляет функцию f_k. Результатом кодирования является КНФ

$$C(f_k, y) = \left(\mathop{\&}_{g \in Circ\,(f_k)} C(g) \right) \cdot y_1^{\sigma_1} \cdot \dots \cdot y_m^{\sigma_m}.$$

Здесь $C(g)$ — КНФ, приписываемая функциональному элементу g схемы $Circ(f_k)$, $y_j, j \in \{1,\ldots,m\}$, — булевы переменные, соответствующие выходам схемы $Circ(f_k)$,

$$y \in range\ f_k : y = (\sigma_1,\ldots,\sigma_m).$$

Используя идеи работы [14], несложно показать, что решив SAT-задачу $C(f_k, y) = 1$, мы найдем такое слово $x \in \{0,1\}^k$, что $f_k(x) = y$.

Далее будем считать, что алгоритм A — это некоторая версия алгоритма DPLL. Следующее утверждение означает, что в этом случае для SAT-задачи, кодирующей задачу обращения дискретной функции f_k, в качестве стартового декомпозиционного множества можно выбрать множество, образованное переменными, которые соответствуют входным полюсам схемы $Circ(f_k)$.

Утверждение 3. *Пусть КНФ $C = C(x_1,\ldots,x_n)$ кодирует задачу обращения некоторой дискретной функции $f_k : \{0,1\}^k \to \{0,1\}^*$. Пусть A — хронологический либо нехронологический алгоритм DPLL. И пусть $\tilde{X}_0$ — декомпозиционное множество, которое образовано переменными из X, соответствующими входным полюсам схемы, представляющей f_k. Тогда значение $F_A\left(C, \tilde{X}_0\right)$ вычислимо за время $O(|C| \cdot N)$.*

Доказательство. Пусть $\tilde{X}_0 = \{x_1,\ldots,x_k\}$ — множество, описанное в условии. Заметим, что в соответствии с выводами работ [15, 16] для произвольного $(\alpha_1,\ldots,\alpha_k) \in \{0,1\}^k$ применение к КНФ вида

$$x_1^{\alpha_1} \cdot \ldots \cdot x_k^{\alpha_k} \cdot C(x_1,\ldots,x_n)$$

правила Unit Propagation [17] и поглощения детерминированным образом приводит либо к выводу выполняющего C набора, либо к выводу конфликта. Использование специальных структур данных [17] дает алгоритм, выполняющий эти действия за время $O(|C|)$. Всего для случайной выборки объема N, сделанной из множества $\{0,1\}^k$, в соответствии с формулой 3.1 потребуется N раз выполнить описанную процедуру распространения ограничений, подсчитать суммарное время, которое на это затрачено, и умножить результат на $2^k/N$. Очевидно, что вычислительные затраты на процедуры умножения и деления, требуют времени $o(|C|)$. Таким образом, общее время, затраченное на вычисление $F_A\left(C, \tilde{X}_0\right)$, есть $O(|C| \cdot N)$. □

Переходим к построению вычислительных алгоритмов минимизации прогнозных функций. При этом мы сталкиваемся с особенностями,

отмеченными выше в пункте 2. Эти особенности дают основание заключить, что наиболее естественной для минимизации $F_A\left(C, \tilde{X}\right)$ является стратегия последовательных улучшений. При этом на начальном шаге строится стартовое декомпозиционное множество $\tilde{X}_0$, такое, что значение $F_A\left(C, \tilde{X}_0\right)$ вычисляется за небольшое время. После этого пытаемся улучшить это значение, рассматривая окрестность «точки» $\tilde{X}_0$ в некотором пространстве поиска. Если это удается, то переходим к новой точке и так далее. Для реализации описанного подхода далее предлагается метод, представляющий собой комбинацию стандартной схемы локального поиска [18] и метаэвристики, в роли которой используется процедура «имитации отжига» [19].

Начнем с определения пространства поиска. Обратим внимание на тот факт, что произвольное множество $\tilde{X} \in 2^X$ можно задать при помощи двоичного вектора

$$\chi\left(\tilde{X}\right) = \left(\chi_1\left(\tilde{X}\right), \ldots, \chi_n\left(\tilde{X}\right)\right),$$

такого, что

$$\chi_i\left(\tilde{X}\right) = \begin{cases} 1, x_i \in \tilde{X} \\ 0, x_i \notin \tilde{X} \end{cases}, i = 1, \ldots, n.$$

Тогда пространство поиска — это n-мерный булев куб $E^n = \{0,1\}^n$. Для произвольной точки $\chi\left(\tilde{X}\right)$ из E^n окрестность $O_r\left(\chi\left(\tilde{X}\right)\right)$ радиуса r определяется как множество всех векторов из E^n, находящихся от $\chi\left(\tilde{X}\right)$ на расстоянии Хэмминга, не превосходящем r.

Стандартная схема локального поиска [18] в применении к задаче минимизации $F_A\left(C, \tilde{X}\right)$ на E^n выглядит следующим образом. Рассматривается точка $\chi\left(\tilde{X}_0\right)$, значение $F_A\left(C, \tilde{X}_0\right)$ в которой вычисляется эффективно. Фиксируем некоторый радиус r и последовательно перебираем точки окрестности $O_r\left(\chi\left(\tilde{X}_0\right)\right)$. Если найдена точка $\chi\left(\tilde{X}_1\right)$: $F_A\left(C, \tilde{X}_1\right) < F_A\left(C, \tilde{X}_0\right)$, то переходим к рассмотрению окрестности $O_r\left(\chi\left(\tilde{X}_1\right)\right)$. Продолжаем процесс до тех пор, пока не найдется точка локального минимума, то есть такая точка $\chi\left(\tilde{X}_s\right)$, что

$$\forall \chi\left(\tilde{X}'\right) \in O_r\left(\chi\left(\tilde{X}_s\right)\right) \Rightarrow F_A\left(C, \tilde{X}'\right) \geq F_A\left(C, \tilde{X}_s\right).$$

Обычно локальный поиск в этом случае останавливается и выдает в качестве ответа точку $\chi\left(\tilde{X}_s\right)$. Для продолжения поиска можно использовать различные подходы. Одним из наиболее эффективных и

простых в реализации является поиск с запретами [19]. Далее предлагается новый метод, который, с нашей точки зрения, наилучшим образом учитывает особенности прогнозных функций. Данный метод является комбинацией локального поиска и метаэвристики, позволяющей выходить из точек локального минимума.

Отметим здесь крайне важную особенность прогнозных функций. Время вычисления значения прогнозной функции в произвольной точке пространства E^n, вообще говоря, не известно заранее. В некоторых точках оно может быть крайне малым, а в других, наоборот, очень большим. Основной вклад в сложность вычисления $F_A\left(C,\tilde{X}\right)$ вносит этап подсчета значений величин $\xi^j = \xi_A\left(\alpha^j_{i_1},\ldots,\alpha^j_{i_d}\right)$, $j = 1,\ldots,N$. Время, требуемое для этого, существенно больше времени, которое требуется для умножения получаемых значений на величину $2^d/N$. Исходя из этого, можно организовать подсчет $F_A\left(C,\tilde{X}\right)$ в виде итерационного процесса, который последовательно вычисляет величины

$$F^j_A\left(C,\tilde{X}\right),\ j = 1,\ldots,N,$$

где

$$F^1_A\left(C,\tilde{X}\right) = \frac{2^d}{N}\cdot\xi^1, F^j_A\left(C,\tilde{X}\right) = F^{j-1}_A\left(C,\tilde{X}\right) + \frac{2^d}{N}\cdot\xi^j, j = 2,\ldots,N. \tag{3.2}$$

Тем самым,

$$F_A\left(C,\tilde{X}\right) = F^N_A\left(C,\tilde{X}\right).$$

Использование соотношений 3.2 позволяет существенно ускорить процесс локального поиска. Действительно, предположим, что локальный поиск стартует с точки $\chi\left(\tilde{X}_0\right)$, значение прогнозной функции в которой равно $F_A\left(C,\tilde{X}_0\right)$. Рассмотрим произвольную точку $\chi\left(\tilde{X}'\right) \in O_r\left(\chi\left(\tilde{X}_0\right)\right)$ и будем подсчитывать значение прогнозной функции в данной точке в соответствии с 3.2. Тогда если для некоторого $j < N$ имеет место

$$F^j_A\left(C,\tilde{X}'\right) > F_A\left(C,\tilde{X}_0\right),$$

то сразу заключаем, что $F_A\left(C,\tilde{X}'\right) > F_A\left(C,\tilde{X}_0\right)$. В этой ситуации можно прервать дальнейшую обработку выборки, соответствующей точке $\chi\left(\tilde{X}'\right)$, и сделать переход к следующей точке окрестности $O_r\left(\chi\left(\tilde{X}_0\right)\right)$.

Основной минус локального поиска состоит в том, что он гарантирует нахождение лишь некоторого локального минимума рассматриваемой функции. Далее предлагается использовать для выхода из точки

локального минимума метаэвристический алгоритм «имитации отжига» [20]. Данный шаг мы оправдываем тем, что функция $F_A\left(C, \tilde{X}\right)$ не задана в виде формулы.

Напомним, что в соответствии со схемой имитации отжига минимизация некоторой функции $\Phi(\cdot)$ рассматривается как итеративный процесс переходов между точками пространства поиска:

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

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

$$\Pr\left\{\tilde{\alpha}^i \to \alpha^{i+1} | \alpha^i\right\} = \begin{cases} 1, \text{ if } \Phi\left(\tilde{\alpha}^i\right) < \Phi\left(\alpha^i\right) \\ \exp\left(-\frac{\Phi\left(\tilde{\alpha}^i\right) - \Phi\left(\alpha^i\right)}{T_i}\right), \text{ if } \Phi\left(\tilde{\alpha}^i\right) \geq \Phi\left(\alpha^i\right) \end{cases} \quad (3.3)$$

Изменение параметра T_i соответствует уменьшению «температуры кристаллизирующейся среды» [20]. 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}.

Метод имитации отжига «в чистом виде» плохо применим к минимизации прогнозных функций в силу отмеченной выше особенности — для использования соотношений 3.3 необходимо досчитывать значение $F_A\left(C, \tilde{X}\right)$ в каждой точке пространства поиска, что сопряжено с существенными вычислительными затратами. Поэтому везде далее мы использовали комбинированную схему, включающую локальный поиск с запоминанием предыстории поиска и симуляцию отжига. Кратко опишем общие принципы данной схемы.

Предполагаем, что на начальном этапе был использован обычный локальный поиск, и его результатом является точка локального минимума $\chi\left(\tilde{X}_s\right)$. Для выхода из $\chi\left(\tilde{X}_s\right)$ применим имитацию отжига. То есть, в соответствии с соотношениями 3.3, сделаем вероятностный переход от $\chi\left(\tilde{X}_s\right)$ к некоторой точке

$$\tilde{X}' \in O_r\left(\chi\left(\tilde{X}_s\right)\right) \ : \ F\left(C, \tilde{X}'\right) \geq F\left(C, \tilde{X}_s\right)$$

(при большом значении температуры соответствующая вероятность велика). Рассмотрим последовательность точек $\chi\left(\tilde{X}_0\right), \ldots, \chi\left(\tilde{X}_t\right)$, полученных в процессе работы описанной схемы. В силу сказанного за-

ключаем, что переход от $\chi\left(\tilde{X}_{t-1}\right)$ к $\chi\left(\tilde{X}_t\right)$ может происходить либо в результате использования схемы локального поиска в окрестности $O_r\left(\chi\left(\tilde{X}_{t-1}\right)\right)$, либо в результате применения процедуры имитации отжига. Выбор точки $\chi\left(\tilde{X}_t\right)$ считаем началом итерации с номером t.

Далее везде будем обозначать через Ψ_{t-1} рекордное значение прогнозной функции, достигнутое на итерациях с номерами $1,\ldots,t-1$. То есть для любой из точек $\chi\left(\tilde{X}_i\right)$, $i \in \{1,\ldots,t-1\}$, имеет место

$$F_A\left(C,\tilde{X}_i\right) \geq \Psi_{t-1}.$$

Определение 3. *Выбранную на итерации с номером t точку $\chi\left(\tilde{X}_t\right)$, такую, что*

$$\forall\chi\left(\tilde{X}'\right) \in O_r\left(\chi\left(\tilde{X}_t\right)\right) \Rightarrow F_A\left(C,\tilde{X}'\right) \geq \min\left\{\Psi_{t-1}, F_A\left(C,\tilde{X}_t\right)\right\},$$

назовем локально неулучшаемой.

Очевидно, что произвольная точка локального минимума является локально неулучшаемой, однако обратное, вообще говоря, неверно (действительно, точка $\chi\left(\tilde{X}_t\right)$ может находиться в окрестности некоторого локального минимума и быть при этом локально неулучшаемой относительно рекорда Ψ_{t-1}).

Рассматриваем итерацию с номером t, $t \geq 1$. Данная итерация начинается с выбора (каким-либо образом) точки $\chi\left(\tilde{X}_t\right)$. Обозначим через S_{loc}^{t-1} список, состоящий из всех локально неулучшаемых точек, пройденных на предыдущих итерациях.

Утверждение 4. *Пусть S_{loc}^{t-1}– список локально неулучшаемых точек, пройденных на первых $t-1$ итерациях, и Ψ_{t-1}– соответствующее рекордное значение прогнозной функции. Тогда для любой точки $\chi\left(\tilde{X}_*\right)$, такой, что $F_A\left(C,\tilde{X}_*\right) < \Psi_{t-1}$, справедливо*

$$\chi\left(\tilde{X}_*\right) \notin \bigcup_{\chi\left(\tilde{X}\right)\in S_{loc}^{t-1}} O_r\left(\chi\left(\tilde{X}\right)\right).$$

Доказательство. Предположим, что точка $\chi\left(\tilde{X}_*\right)$ с указанными свойствами принадлежит окрестности $O_r\left(\chi\left(\tilde{X}\right)\right)$ некоторой точки $\chi\left(\tilde{X}\right) \in S_{loc}^{t-1}$. Но для любой такой точки имеет место $F_A\left(C,\tilde{X}\right) \geq \Psi_{t-1}$ в силу определения неулучшаемости. Имеем противоречие. □

Данный факт позволяет значительно сократить время работы алгоритма за счет того, что значение прогнозной функции не имеет смысла вычислять в окрестностях радиуса r неулучшаемых точек.

Итак, с учетом всего сказанного описываем схему минимизации прогнозных функций. На начальной итерации выбирается точка $\chi\left(\tilde{X}_0\right)$, значение прогнозной функции в которой вычисляется эффективно, при этом полагаем $\Psi_0 = F_A\left(C, \tilde{X}_0\right)$. Рассматриваем итерацию с номером t ($t \geq 1$) и пусть $\chi\left(\tilde{X}_t\right)$ – выбранная в начале данной итерации точка. Пусть Ψ_{t-1} — рекорд, достигнутый на предыдущих итерациях. Рассмотрим множество

$$\Omega\left(\chi\left(\tilde{X}_t\right)\right) = O_r\left(\chi\left(\tilde{X}_t\right)\right) \setminus \left(\bigcup_{\chi(\tilde{X}) \in S_{loc}^{t-1}} O_r\left(\chi\left(\tilde{X}\right)\right) \right).$$

Если существует точка $\chi\left(\tilde{X}'\right) \in \Omega\left(\chi\left(\tilde{X}_t\right)\right)$, такая, что

$$F_A\left(C, \tilde{X}'\right) < \min\left\{ \Psi_{t-1},\ F_A\left(C, \tilde{X}_t\right) \right\},$$

то полагаем $\chi\left(\tilde{X}_{t+1}\right) = \chi\left(\tilde{X}'\right)$, $\Psi_t = \min\left\{ \Psi_{t-1}\ ,\ F_A\left(C, \tilde{X}_t\right) \right\}$, $S_{loc}^t = S_{loc}^{t-1}$. После этого переходим к итерации с номером $t+1$, на которой рассматриваем точку $\chi\left(\tilde{X}_{t+1}\right)$. Если же такой точки не существует, то объявляем $\chi\left(\tilde{X}_t\right)$ локально неулучшаемой. После этого переходим от точки $\chi\left(\tilde{X}_t\right)$ к точке

$$\chi\left(\tilde{X}_{t+1}\right) \in O_r\left(\chi\left(\tilde{X}_t\right)\right) \setminus S_{loc}^{t-1},$$

используя имитацию отжига. Полагаем $\Psi_t = \min\left\{\Psi_{t-1}\,,\ F_A\left(C, \tilde{X}_t\right)\right\}$ и $S_{loc}^t = S_{loc}^{t-1} \bigcup \left\{\chi\left(\tilde{X}_t\right)\right\}$.

При каждом вызове процедуры имитации отжига температура понижается. Процесс поиска останавливается, когда необходим запуск процедуры имитации отжига, но температура и, следовательно, вероятность перехода 3.3 близки к 0.

4. Заключение

Описанные выше алгоритмы были реализованы на вычислительном кластере ИДСТУ СО РАН [21]. В качестве тестовой рассматривалась

задача построения декомпозиционного множества для SAT-задачи, кодирующей криптоанализ генератора A5/1 [22]. В работе [23] было приведено декомпозиционное множество, построенное «вручную» с учетом особенностей алгоритма данного генератора. Это множество изображено на рисунке 1.

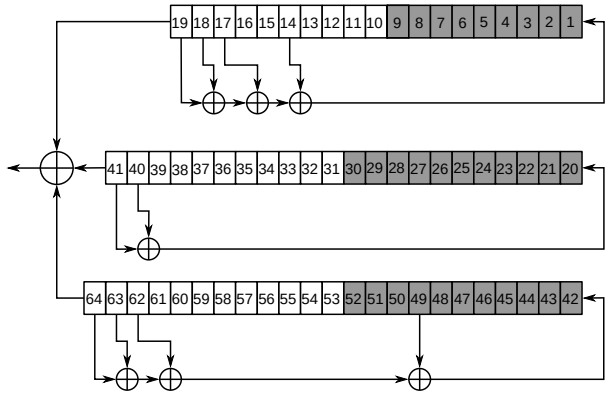

Рис. 1. Декомпозиционное множество, структура которого обусловлена особенностями алгоритма генератора A5/1 [23].

Затем для поиска декомпозиционного множества в рассматриваемой SAT-задаче были применены описанные выше алгоритмы. В качестве стартового декомпозиционного множества выбиралось множество, образованное 64 переменными, кодирующими вход булевой функции, реализующей рассматриваемый генератор. В процессе минимизации прогнозной функции выбирались декомпозиционные множества, являющиеся подмножествами стартового множества. Результат поиска изображен на рисунке 2.

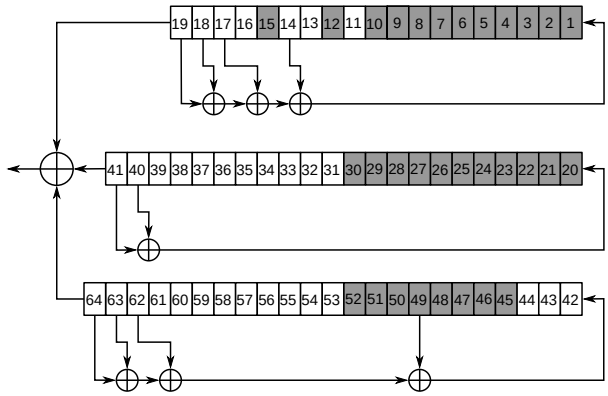

Рис. 2. Декомпозиционное множество, найденное автоматически при помощи алгоритмов, представленных в работе.

Похожие результаты были получены для более простых генераторов (порогового и суммирующего) – декомпозиционные множества, построенные описанными выше алгоритмами, незначительно отличались от известных «эталонных» множеств. На основании сказанного можно заключить, что предложенные в работе алгоритмы позволяют успешно выявлять структуру декомпозиционных множеств в SAT-задачах, кодирующих некоторые трудные комбинаторные проблемы.

Список литературы

1. Handbook of Satisfiability / A. Biere, V. Heule, H. van Maaren, T. Walsh. — IOS Press, 2009. – 980 p.
2. Up-to-date links for the SATisfiability problem [Electronic recource]: www.satlive.org
3. Schubert T. PaMiraXT: Parallel SAT Solving with Threads and Message Passing / T. Schubert, M. Lewis, B. Becker // Journal on Satisfiability, Boolean Modeling and Computation. – 2009. – Vol. 6. – P. 203–222.
4. Hamadi Y. ManySAT: a Parallel SAT Solver / Y. Hamadi, S. Jabbour, L. Sais // Journal on Satisfiability, Boolean Modeling and Computation. – 2009. – Vol. 6. – P. 245–262.
5. Ignatiev A. DPLL+ROBDD Derivation Applied to Inversion of Some Cryptographic Functions / A. Ignatiev, A. Semenov // LNCS. – 2011. – Vol. 6695. – P. 76–89.
6. Schulz S. Parallel SAT Solving on Peer-to-Peer Desktop Grids / S. Schulz, W. Blochinger // Journal Of Grid Computing. – 2010. – Vol. 8, N 3. – P. 443–471.
7. Hyvarinen A. Grid-Based SAT Solving with Iterative Partitioning and Clause Learning / A. Hyvarinen, I. Niemela, T. Junttila // LNCS. – 2011. – Vol. 6876. – P. 385–399.
8. Posypkin M. Using BOINC desktop grid to solve large scale SAT problems / M. Posypkin, A. Semenov, O. Zaikin // Computer Science Journal. — 2012. – Vol. 13, N 1. – P. 25–34.
9. Davis M. A machine program for theorem proving / M. Davis, G. Logemann, D. Loveland // Communication of the ACM. – 1962. – Vol. 5, issue 7. – P. 394–397.
10. Marques-Silva J. P. GRASP: A search algorithm for propositional satisfiability / J. P. Marques-Silva, K. A. Sakallah // IEEE Transactions on Computers. – 1999. – Vol. 48, N 5. – P. 506–521.
11. Metropolis N. The Monte Carlo method / N. Metropolis, S. Ulam // J. Amer. statistical assoc. – 1946. – Vol. 44, № 247. – P. 335–341.
12. Ермаков С. М. Метод Монте-Карло и смежные вопросы / С. М. Ермаков. – М. : Наука, 1971. – 327 с.
13. Стрекаловский А. С. Элементы невыпуклой оптимизации / А. С. Стрекаловский. – Новосибирск : Наука, 2003. 355 с.
14. Цейтин Г. С. О сложности вывода в исчислении высказываний / Г. С. Цейтин // Записки научных семинаров ЛОМИ АН СССР. — 1968. — Т. 8. — С. 234–259.
15. Jarvisalo M. Limitations of restricted branching in clause learning / M. Jarvisalo, T. Junttila // Constraints. — 2009. — Vol. 14, № 3. — pp. 325–356.
16. Семенов А. А. Декомпозиционные представления логических уравнений в задачах обращения дискретных функций / А. А. Семенов // Изв. РАН. Теория и системы управления. – 2009. – № 5. – С. 47–61.

17. Dowling W. Linear-time algorithms for testing the satisfiability of propositional Horn formulae / W. Dowling, J. Gallier // Journal of Logic Programming. – 1984. – Vol. 1, N 3. – P. 267–284.
18. Пападимитриу Х. Комбинаторная оптимизация / Х. Пападимитриу, К. Стайглиц. – М. : Мир, 1985. – 510 с.
19. Glover F. Tabu Search / F. Glover, M. Laguna. – Dordrecht : Kluwer Acad. Publ., 1997.
20. Kirkpatrick S. Optimization by simulated annealing / S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi // Science. – 1983. – Vol. 220. – P. 671–680.
21. Кластер Blackford ИДСТУ СО РАН: http://hpc.icc.ru/hardware/blackford.php
22. Biryukov A. Real time cryptanalysis of A5/1 on a PC / A. Biryukov, A. Shamir, D. Wagner // LNCS. – 2001. – Vol. 1978. – P. 1–18.
23. Parallel logical cryptanalysis of the generator A5/1 in BNB-Grid system / A. Semenov, O. Zaikin, D. Bespalov, M. Posypkin // LNCS. – 2011. – Vol. 6873. – P. 473–483.

A. A. Semenov, O. S. Zaikin

Algorithms for constructing decomposition sets in application to coarse-grained parallelization of SAT problems.

Abstract. In the paper algorithms for constructing decomposition sets in application to coarse-grained parallelization of SAT problems are suggested. These sets are used for solving SAT problems in distributed computing environments. Suggested algorithms are based on computing scheme of Monte-Carlo method.

Keywords: Monte-Carlo method; discrete functions; SAT problems; coarse-grained parallelization.

Семенов Александр Анатольевич, кандидат технических наук, зав. лабораторией дискретного анализа и прикладной логики, Институт динамики систем и теории управления СО РАН, 664033, Иркутск, ул. Лермонтова, 134, тел.: (3952)453054 (biclop@rambler.ru)

Заикин Олег Сергеевич, кандидат технических наук, научный сотрудник лаборатории дискретного анализа и прикладной логики, Институт динамики систем и теории управления СО РАН, 664033, Иркутск, ул. Лермонтова, 134, тел.: (3952)453054 (zaikin.icc@gmail.com)

Semenov Alexander, Institute for System Dynamics and Control Theory of SB RAS, Lermontov str., 134, Post Box 292 664033, Irkutsk, Russia Phone: (3952)453054 (biclop.rambler@yandex.ru)

Zaikin Oleg, Institute for System Dynamics and Control Theory of SB RAS, Lermontov str., 134, Post Box 292 664033, Irkutsk, Russia Phone: (3952)453054 (zaikin.icc@gmail.com)